

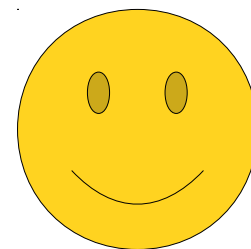


**A tutorial introduction to
the QT Creator debugger**

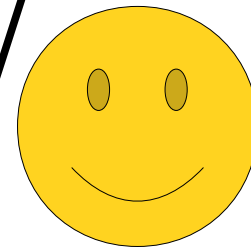
written by Keith Schwarz

Assignment 0: Using the Debugger

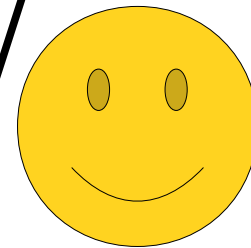
Hi everybody!



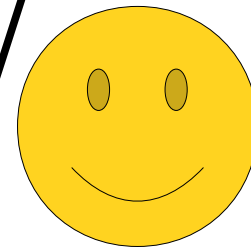
As part of Assignment 0, we'd like you to get a little bit of practice using the debugger in Qt Creator.



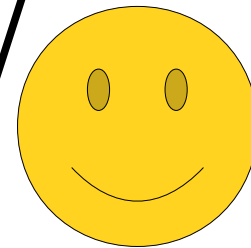
The debugger is a tool you can use to help see what your program is doing as you run it.



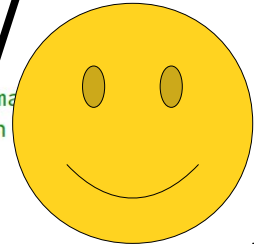
It's really useful for helping find errors in your programs, and the more practice you get with it, the easier it'll be to correct mistakes in the programs you write.



Think of this guide as a little tutorial walkthrough to help give you a sense of how to use the debugger and how to make sense of what you're seeing.



To start things off, open up the Name Hash program you ran in Part One of this assignment. Scroll down to the nameHash function so that you can see the entire function in your window.



```
39 * th
40 * of
41 *
42 * Fo
43 * treat
44 * It then uses them as coefficients in a polynomial over
45 *  $F_p$ , where  $p$  is a large prime number, and evaluates the
46 * some smaller prime number  $q$ . (You aren't expected to know  $q$ ,
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Type to locate (Ctrl...

1 Issues

2 Search Results

3 Application Output

4 Compile Output

5 QML Debugger Console

6 General Messages

8 Test Results

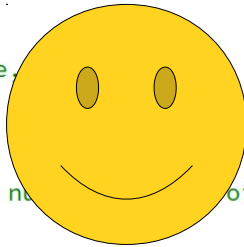
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
```

Move your mouse cursor so that it's in the space right before the line number for line 66.

Now, click the mouse!



```
55 static const int kSmallPrime =
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The number of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```



Projects

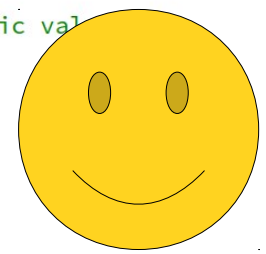
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
```

When you do, you should see a red circle with a little hourglass pop up.

This is called a **breakpoint**. If we run the program in debug mode, whenever the program gets to this line, it will pause and open up the debugger so we can see what's going on.

```
60 // name, updating the hash at each
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case.
64      * lower-case letters are always less than 127.
65      */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```



Qt

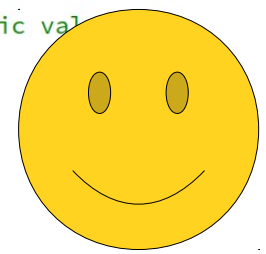
Debug

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS106B,
47 * but we thought it might be fun!)
48 */
```

Now, we're going to run this program in debug mode. To do so, click on the "run in debug mode" button in the bottom-right corner of the screen. It's the one just below the regular green "run" button. When you do...



```
60 // name, updating the hash at each
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case.
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Qt

Debug

Run in debug mode button (blue bug icon)



... you should see something like this! Notice that a bunch of extra panels popped up in Qt Creator. We'll talk about what each of these windows mean in a second.

The screenshot shows the Qt Creator IDE interface. The main editor displays the following C++ code:

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31.
53  */
54  stat
55  stat
56
57  int
58
59  /* I
60  * n
61  */
62  for
63
64
65
66
67
68  }
69  retu
70  }
71
```

The console window shows the prompt: "What is your first name?". A yellow smiley face is overlaid on the console.

The debugger window at the bottom shows the following table:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring)	...eHash.cpp	66					...555892674			(all)

The bottom status bar shows the following tabs: 1 Issues, 2 Search Results, 3 Application Output, 4 Compile Output, 5 QML Debugger Console, 6 General Messages, 8 Test Results.

In the meantime, type in the first name **Ada** and hit enter, as shown here.

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31.
53  */
54  stat
55  stat
56  What is your first name? Ada
57  int What is your last name?
58
59  /* I
60  * n
61  */
62  for
63
64
65
66
67
68  }
69  retu
70  }
71
```

Name	Value	Type

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1eHash.cpp	66	66	...555892674			(all)



Now, type in **Lovelace** as a last name, but
don't hit enter yet!

The screenshot shows the Qt IDE interface. The main editor displays the following C++ code:

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31.
53     */
54     stat
55     stat
56     int
57     int
58     /* I
59     * n
60     */
61     */
62     for
63     for
64
65
66
67
68 }
69 return
70 }
71
```

The console window shows the program's output:

```
File Edit Options Help
What is your first name? Ada
What is your last name? Lovelace
```

A yellow smiley face is overlaid on the console. The debugger window at the bottom shows the application has started at line 66 of NameHash.cpp.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1eHash.cpp	66555892674	(all)

As soon as you hit enter, a bunch of things are going to pop up in Qt Creator. Don't panic! It's normal.

The screenshot shows the Qt Creator IDE with a C++ project named "Welcome to CS106B". The main editor displays the following code:

```
48  */
49  int nameHash(string first, string last){
50      /* This hashing scheme needs two prime numbers, a large prime and a small
51       * prime. These numbers were chosen because their product is less than
52       * 2^31.
53       */
54      stat
55      stat
56      int
57      int
58      /* I
59       * n
60       */
61      */
62      for
63      for
64
65
66
67
68      }
69      retu
70  }
71
```

The console window shows the program's output:

```
File Edit Options Help
What is your first name? Ada
What is your last name? Lovelace
```

A yellow smiley face is drawn on the right side of the console window.

The debugger window at the bottom shows the following table:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1ring)	...	66eHash.cpp	66	...	555892674		(all)

The bottom status bar shows the following tabs: 1 Issues, 2 Search Results, 3 Application Output, 4 Compile Output, 5 QML Debugger Console, 6 General Messages, 8 Test Results.

With that said, hit enter,
and watch the magic happen!

The screenshot shows the Qt IDE interface. The main editor displays the following C++ code:

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31.
53     */
54     stat
55     stat
56
57     int
58     What is your first name? Ada
59     What is your last name? Lovelace
60
61     /* I
62     * n
63     */
64     for
65
66
67
68 }
69
70
71
```

The console window shows the following output:

```
File Edit Options Help
What is your first name? Ada
What is your last name? Lovelace
```

A yellow smiley face is overlaid on the console output.

The debugger window shows the following table:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1eHash.cpp	66	66	...555892674			(all)

The bottom status bar shows the following tabs: 1 Issues, 2 Search Results, 3 Application Output, 4 Compile Output, 5 QML Debugger Console, 6 General Messages, 8 Test Results.

Shazam! We're back in Qt Creator, and there's tons of values showing up everywhere.



```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

There's a lot going on right here. Let's see what's happening.



```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkIn...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```

47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.

```

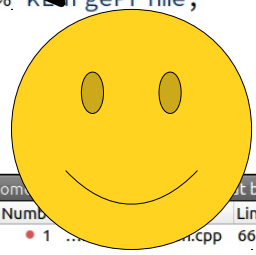
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

First, notice that our red breakpoint now has a yellow arrow in it.

```

63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are all between 'a' and 'z', so we can use the
65     * difference between 'a' and 'A' to convert.
66     */
67     ch = tolower(ch);
68     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69 }
70 return hashVal;
71 }

```



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome to CS106B

Level	Function	File	Line	Number	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	66	...			(all)
2	qMain	Nam...	31						
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302						
4	std::function<int (&)>::operator()() const	std_f...	706						
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981						
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302						
7	std::function<int (&)>::operator()() const	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514						
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::invoke_o...	invok...	60						

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

This yellow arrow indicates where in the program we are right now. The program stopped running at this line because we hit that breakpoint you set earlier.

```
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are adjacent to each other, so we can use
65     * arithmetic to convert them to lower case.
66     */
67     ch = tolower(ch);
68     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69 }
70 return hashVal;
71 }
```



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome to CS106B

Level	Function	File	Line	Number	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	66	...555892674			(all)
2	qMain	Nam...	31						
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302						
4	std::function<int (&)>::operator()() const	std_f...	706						
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981						
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThreadSt...	std_f...	302						
7	std::function<int (&)>::operator()() const	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514						
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::invo...	invok...	60						

the last

Qt Creator sidebar icons: Welcome, Edit, Debug, Projects, Help, Qt logo, and system tray icons.

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```

47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.

```

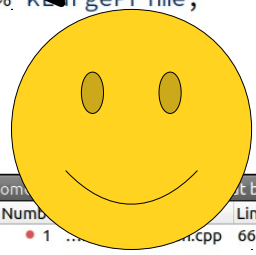
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Whenever you pop up the debugger, it's good to figure out exactly where you are in the program that you're running, so you'll get into the habit of checking for this yellow arrow.

```

63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are adjacent.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71

```



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome to CS106B

Level	Function	File	Line	Number	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	66	...			(all)
2	qMain	Nam...	31						
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302						
4	std::function<int (&)>::operator()() const	std_f...	706						
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981						
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThreadStd::run)	std_f...	302						
7	std::function<int (&)>::operator()() const	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514						
10	std::invoke_impl<void, GThreadStd::start()>::<lambda>>(std::_invoke_o...	invok...	60						

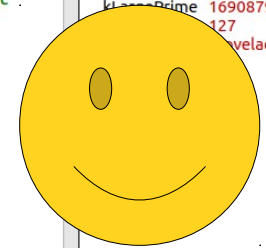
File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64     for (ch : last)
65         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
66     return hashVal;
67 }
68
69
70
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kSmallPrime	127
kLargePrime	16908799
last	"Lovelace"



Next, let's take a look at this panel.
This is called the **call stack**.

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

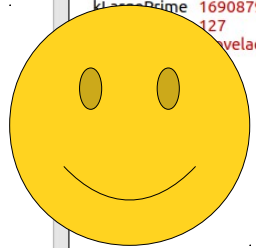
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::invo...	invok...	60								

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

File Edit Build Debug Analyze Tools Window Help

Projects
Welcome to CS106B [master]
Welcome to CS106B.pro
Headers
Sources
lib
res
src
NameHash.cpp
Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64     for (char ch : last)
65         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
66     return hashVal;
67 }
68
69
70
71 }
```



Right now, we know we're in the nameHash function, because our helpful friend the Yellow Arrow tells us exactly what line we're on!

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()(void) const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()(void) const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

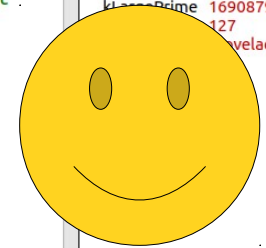
Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64
65     for (char ch : last)
66         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
67
68     return hashVal;
69 }
70
71 }
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
lovelace	"Lovelace"



However, the yellow arrow can't tell us exactly how we got to this part of the program. What part of the program actually called nameHash?

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()(void) const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()(void) const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

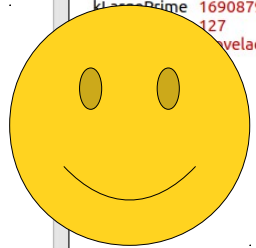
File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64
65     for (char ch : last)
66         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
67
68     return hashVal;
69 }
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
lovelace	"Lovelace"



The call stack can tell us exactly that!

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

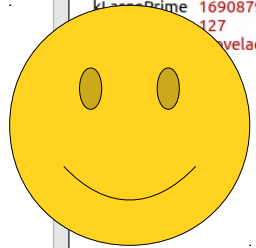
File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64     for (ch : last)
65         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
66     return hashVal;
67 }
68
69
70
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
lovelace	"Lovelace"



Notice that the call stack lists a series of different functions in order. Here, it has nameHash (where we are now) at the top, and right below that is qMain.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...(ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

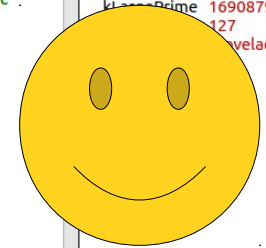
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Qt Welcome Edit Debug Projects Help

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch : first)
63         hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;
64     for (char ch : last)
65         hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;
66     return hashVal;
67 }
68
69
70
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kSmallPrime	16908799
kLargePrime	127
lovelace	"Lovelace"



Go and double-click the call to Main on Level 1. When you do...

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	1	nameHash	NameHash.cpp	66	...555892674			(all)
2	qMain	main.cpp	31								
3	std::Function_handler<int (*)>::M_invoke	std_function_handler.h	302								
4	std::function<int (*)>::operator()	std_function.h	706								
5	QtGui::<lambda>::operator()	spl.cpp	20981								
6	std::Function_handler<int (*)>::operator()	std_function.h	302								
7	std::function<int (*)>::operator()	std_function.h	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::lambda	std_invoke.h	60								

Qt Welcome Edit Debug Projects Help

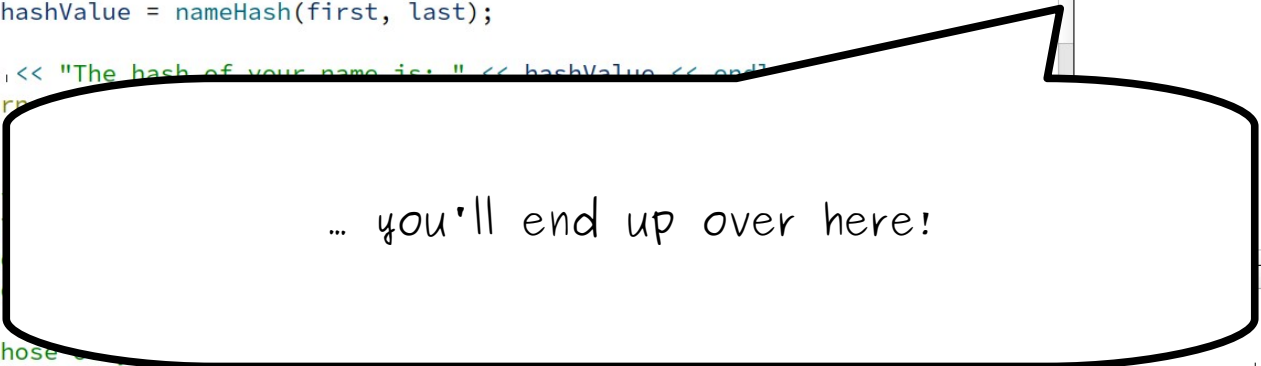
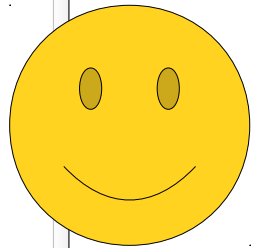
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```

19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This
38  * to ta
39  * the m
40  * of th
41  *
42  * For those
43  * treats each character in the input name as a number between 0 and 128.
44  * It then uses them as coefficients in a polynomial over the finite field

```



Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkIn...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda>>::std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

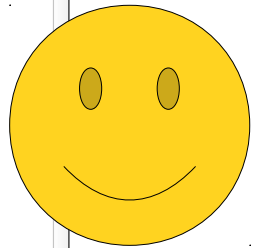
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```

19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This
38  * to ta
39  * the m
40  * of th
41  *
42  * For those
43  * treats each character in the input name as a number between 0 and 128.
44  * It then uses them as coefficients in a polynomial over the finite field

```



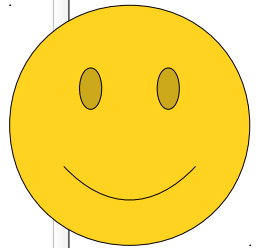
Notice that the highlighted line here includes a call to the nameHash function. This the part of the code that actually called nameHash, which is how we got to the line with the breakpoint!

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (>, int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>>(std::invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This
38 * to ta
39 * the m
40 * of th
41 *
42 * For those
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```



Generally speaking, you can use the call stack as a way to see which function calls got us to the point where the program paused at the breakpoint!

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda>>::std::invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring)	...eHash.cpp	66	...555892674			(all)

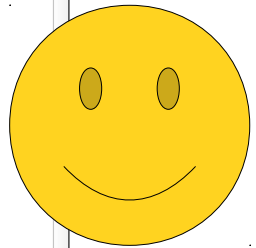
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```

19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This
38  * to ta
39  * the m
40  * of th
41  *
42  * For those
43  * treats each character in the input name as a number between 0 and 128.
44  * It then uses them as coefficients in a polynomial over the finite field

```



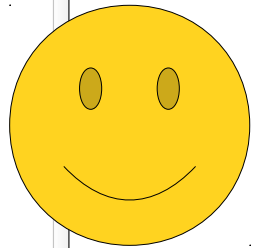
You might notice that there's some more stuff in the call stack beyond just main and nameHash. What are those?

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This
38 * to ta
39 * the m
40 * of th
41 *
42 * For those
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```



Let's find out! Double-click on the function on Level 3. (Here's what it looks like on my system; you might see something different.)

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&), int (*)>::operator() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThreadStd::run	std_f...	302								
7	std::function<int (&), int (*)>::operator() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl_void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Qt

- Welcome
- Edit
- Debug
- Projects
- Help

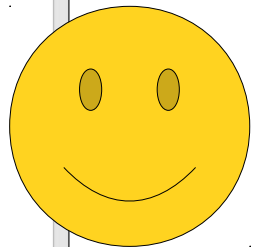
WEL...106B

Debug

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example. [Show Details](#) [Minimize](#)

```

290
291 template<typename _Res, typename _Functor, typename... _ArgTypes>
292 class _Function_handler<_Res(_ArgTypes...), _Functor>
293 : public _Function_base::_Base_manager<_Functor>
294 {
295     typedef _Function_base::_Base_manager<_Functor> _Base;
296
297 public:
298     static _Res
299     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
300     {
301         return (*_Base::_M_get_pointer(__functor))(
302             std::forward<_ArgTypes>(__args)...);
303     }
304 };
305
306 template<typename _Functor, typename... _ArgTypes>
307 class _Function_handler<void(_ArgTypes...), _Functor>
308 : public _Function_base::_Base_manager<_Functor>
309 {
310     typedef _Function_base::_Base_manager<_Functor> _Base;
311
312 public:
313     static void
314     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
  
```



When you do, you'll see something like this.
 (This might be different depending on your OS.
 Don't panic if it doesn't exactly match.)

Level	Function	File	Line
1	nameHash		
2	qMain		
3	std::Function_handler<int(), int (*)>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int(), QtGui::startBackgroundEventLoop(GThunkIn...)	std_f...	302
7	std::function<int ()>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda>>::std::_invoke_o...	invok...	60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Qt

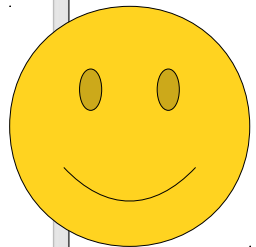
- Welcome
- Edit
- Debug
- Projects
- Help

Qt

Debug

```

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.
290
291 template<typename _Res, typename _Functor, typename... _ArgTypes>
292 class _Function_handler<_Res(_ArgTypes...), _Functor>
293 : public _Function_base::_Base_manager<_Functor>
294 {
295     typedef _Function_base::_Base_manager<_Functor> _Base;
296
297 public:
298     static _Res
299     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
300     {
301         return (*_Base::_M_get_pointer(__functor))(
302             std::forward<_ArgTypes>(__args)...);
303     }
304 };
305
306 template<typename _Functor, typename... _ArgTypes>
307 class _Function_handler<void(_ArgTypes...), _Functor>
308 : public _Function_base::_Base_manager<_Functor>
309 {
310     typedef _Function_base::_Base_manager<_Functor> _Base;
311
312 public:
313     static void
314     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
  
```



Whenever you start up a program in CS106B, there's a little bit of code that we automatically call for you, which does things like setting up the console.

Level	Function	File	Line
1	nameHash		
2	qMain		
3	std::Function_handler<int(), int (*)>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int(), QtGui::startBackgroundEventLoop(GThunkIn...	std_f...	302
7	std::function<int ()>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda>>::std::_invoke_o...	invok...	60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Qt

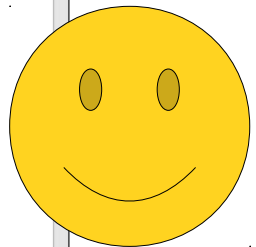
- Welcome
- Edit
- Debug
- Projects
- Help

Qt

Debug

```

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.
290
291 template<typename _Res, typename _Functor, typename... _ArgTypes>
292 class _Function_handler<_Res(_ArgTypes...), _Functor>
293 : public _Function_base::_Base_manager<_Functor>
294 {
295     typedef _Function_base::_Base_manager<_Functor> _Base;
296
297 public:
298     static _Res
299     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
300     {
301         return (*_Base::_M_get_pointer(__functor))(
302             std::forward<_ArgTypes>(__args)...);
303     }
304 };
305
306 template<typename _Functor, typename... _ArgTypes>
307 class _Function_handler<void(_ArgTypes...), _Functor>
308 : public _Function_base::_Base_manager<_Functor>
309 {
310     typedef _Function_base::_Base_manager<_Functor> _Base;
311
312 public:
313     static void
314     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
  
```



This code will show up in the call stack below your actual program.

Level	Function	Address
1	nameHash	
2	qMain	
3	std::Function_handler<int(), int (*)>::_M_invoke(std::Any_data const&)	std f... 302
4	std::function<int ()>::operator()() const	std f... 706
5	QtGui::<lambda>::operator()(void) const	spl.cpp 20981
6	std::Function_handler<int(), QtGui::startBackgroundEventLoop(GThinkn...)	std f... 302
7	std::function<int ()>::operator()() const	std f... 706
8	GThreadStd::run	spl.cpp 22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp 22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok... 60

Projects

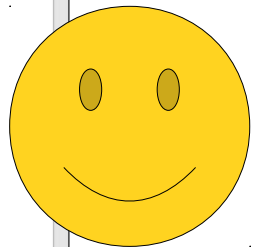
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Debugger GDB for "Wel..."

Level	Function	File	Line
1	nameHash		
2	qMain		
3	std::Function_handler<int (&), int (*)>::M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>std::invoke_o...	invok...	60

```

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.
290
291 template<typename _Res, typename _Functor, typename... _ArgTypes>
292 class _Function_handler<_Res(_ArgTypes...), _Functor>
293 : public _Function_base::_Base_manager<_Functor>
294 {
295     typedef _Function_base::_Base_manager<_Functor> _Base;
296
297 public:
298     static _Res
299     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
300     {
301         return (*_Base::_M_get_pointer(__functor))(
302             std::forward<_ArgTypes>(__args)...);
303     }
304 };
305
306 template<typename _Functor, typename... _ArgTypes>
307 class _Function_handler<void(_ArgTypes...), _Functor>
308 : public _Function_base::_Base_manager<_Functor>
309 {
310     typedef _Function_base::_Base_manager<_Functor> _Base;
311
312 public:
313     static void
314     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
  
```



You shouldn't need to dig around this deep in the call stack, and if you do, it should probably be a message telling you to back up a bit back to code that you actually wrote.

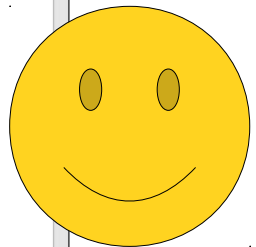
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.
290
291 template<typename _Res, typename _Functor, typename... _ArgTypes>
292 class _Function_handler<_Res(_ArgTypes...), _Functor>
293 : public _Function_base::_Base_manager<_Functor>
294 {
295     typedef _Function_base::_Base_manager<_Functor> _Base;
296
297 public:
298     static _Res
299     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
300     {
301         return (*_Base::_M_get_pointer(__functor))(
302             std::forward<_ArgTypes>(__args)...);
303     }
304 };
305
306 template<typename _Functor, typename... _ArgTypes>
307 class _Function_handler<void(_ArgTypes...), _Functor>
308 : public _Function_base::_Base_manager<_Functor>
309 {
310
311
312
313
314

```



so let's jump back to the code that we actually wrote.

Level	Function	File	Line
1	nameHash		
2	qMain		
3	std::Function_handler<int (), int (*)>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()() const	std_f...	706
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302
7	std::function<int ()>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>::std::invoke_o...	invok...	60

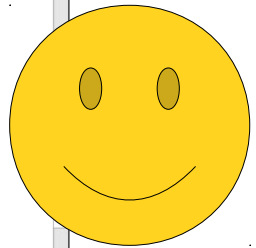
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.
290
291 template<typename _Res, typename _Functor, typename... _ArgTypes>
292 class _Function_handler<_Res(_ArgTypes...), _Functor>
293 : public _Function_base::_Base_manager<_Functor>
294 {
295     typedef _Function_base::_Base_manager<_Functor> _Base;
296
297 public:
298     static _Res
299     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
300     {
301         return (*_Base::_M_get_pointer(__functor))(
302             std::forward<_ArgTypes>(__args)...);
303     };
304 };
305
306 templ
307 cla
308 : p
309 {
310 t
311
312 public:
313     static void
314     _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)

```



To do that, double-click on Level 0, the call to nameHash. When you do...

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1eHash.cpp	66	...555892674			(all)
2	Main	Nam...	31								
3	std::Function_handler<int (), int (*)>::_M_invoke(std::Any_data const&)	std f...	302								
4	std::function<int ()>::operator()() const	std f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkn...	std f...	302								
7	std::function<int ()>::operator()() const	std f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>std::_invoke_o...	invok...	60								

Projects

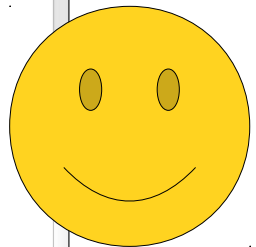
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the last
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67
68  }
69  return hashVal;
70  }
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



You'll be teleported back to safety!

Debugger GDB for "Welc...

Level	Function	File	Line
1	nameHash	NameHash.cpp	66
2	qMain	main.cpp	51
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...>::operator()() const	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

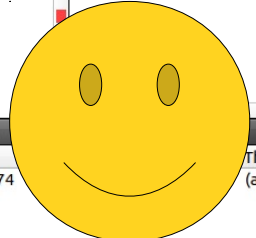
```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /*
60
61
62  f
63
64
65
66  hash
67
68  }
69  return hashVal;
70
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Let's quickly recap what we've seen so far.



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda>>(std::_invoke_o...	invok...	60

Number	Function	File	Line	Address
1	...ring)	...eHash.cpp	66	...555892674

Debugger

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

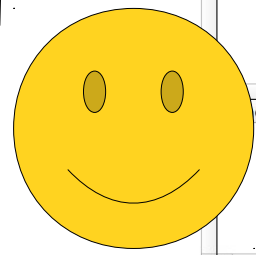
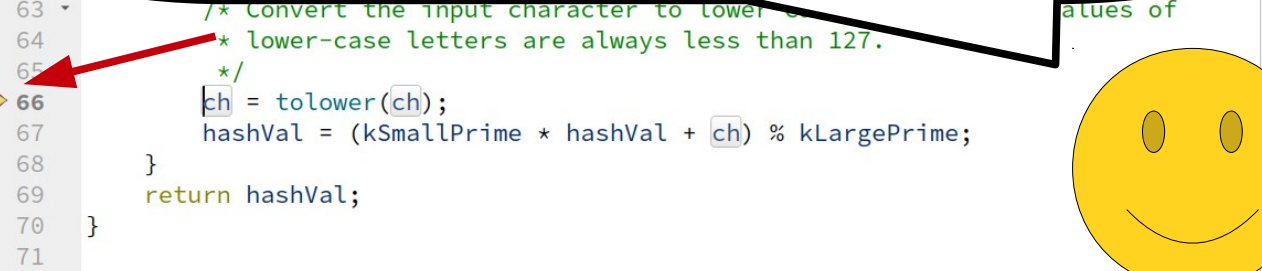
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50      /* This hashing scheme needs two prime numbers, a large prime and a small
51       * prime. These numbers were chosen because their product is less than
52       * 2^31 - kLargePrime - 1.
53       */
54
62
63      /* Convert the input character to lower case. The values of
64       * lower-case letters are always less than 127.
65       */
66      ch = tolower(ch);
67      hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

To set a breakpoint so that we can pause the program and look around, click in the margin just before the line number where you want to pause.



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Debugger

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

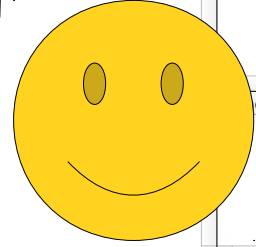
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54
62
63  /* Convert the input character to lower case. The values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

Once the breakpoint is reached, it will pull up all sorts of useful information.



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...>::operator()() const	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Debugger

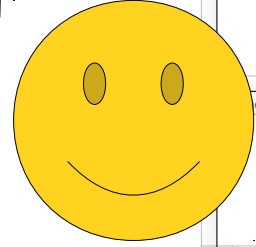
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54
62
63  /* Convert the input character to lower case. The values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71
```

The yellow arrow points out where we are right now.



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring)	...eHash.cpp	66	...555892674			(all)

Qt

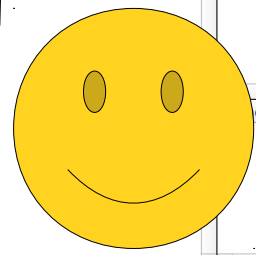
Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
62
63     /* Convert the input character to lower case. The values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

The call stack shows us how we got into the current function.



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (&)*>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()() const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()() const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start():<lambda>>(std::invoke_o...	invok...	60								

Projects

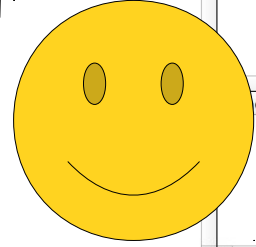
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54
62
63  /* Convert the input character to lower case. The values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

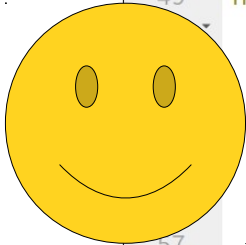
Now, let's see how we can read the values of the variables in this function.



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Look up at this panel over here.



```
47 int hashVal = 0;
48 */
49 int hashVal = 0;
50 int hashVal = 0;
51 int hashVal = 0;
52 int hashVal = 0;
53 int hashVal = 0;
54 int hashVal = 0;
55 int hashVal = 0;
56 int hashVal = 0;
57 int hashVal = 0;
58 int hashVal = 0;
59 */
60 */
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }
```

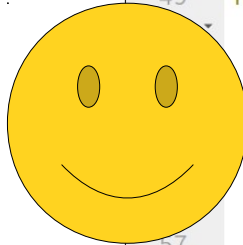
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

This window lets you take a look at all the values of the local variables that are in scope right now.



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

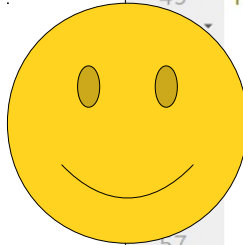
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Depending on what OS you're using, these might be in a different order, and there might be some weird-looking ones in there in addition to nicer ones like ch and hashVal.



```
47 int nameHash(string first, string last){
48     /* This hashing scheme needs two prime numbers, a large prime and a small
49     * prime. These numbers were chosen because their product is less than
50     * 2^31 - kLargePrime - 1.
51     */
52     static const int kLargePrime = 16908799;
53     static const int kSmallPrime = 127;
54
55     int hashVal = 0;
56
57     /* Iterate across all the characters in the first name, then the last
58     * name, updating the hash at each step.
59     */
60     for (char ch: first + last) {
61         /* Convert the input character to lower case. The numeric values of
62         * lower-case letters are always less than 127.
63         */
64         ch = tolower(ch);
65         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
66     }
67     return hashVal;
68 }
69
70
71
```

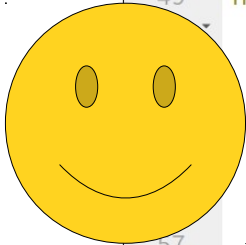
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::<lambda>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

If we ignore the weird-looking ones, we can see some nice, familiar names.



```
int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values of
    * lower-case letters are always less than 127.
    */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
}
```

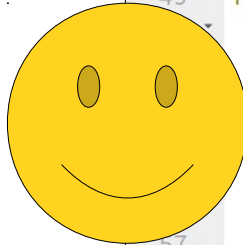
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

For example, here you can see the values of kLargePrime and kSmallPrime, which match the values they were declared with.



```
int nameHash(const string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a small
    * prime. These numbers were chosen because their product is less than
    * 2^31 - kLargePrime - 1.
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

    /* Iterate across all the characters in the first name, then the last
    * name, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric values of
        * lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

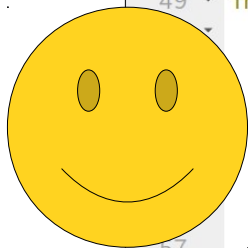
Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (>, int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkIn...	std_f...	302
7	std::function<int (>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring) ...eHash.cpp	66	...	555892674			(all)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

We can also see that, at this point, hashVal is still zero.



```
int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values of
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
}
```



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

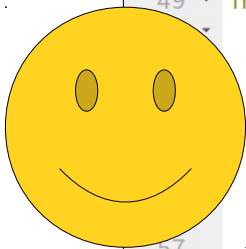
Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (>, int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThun...	std_f...	302
7	std::function<int (>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start():<lambda>>(std::invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring) ...eHash.cpp	66		...555892674			(all)

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

As we walk through the program one step at a time, we'll see these values change.



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

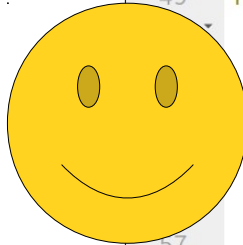
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (>, int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::invoke_o...	invok...	60								

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Now, let's take a look at this for loop.



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

```
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values of
    * lower-case letters are always less than 127.
    */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
```

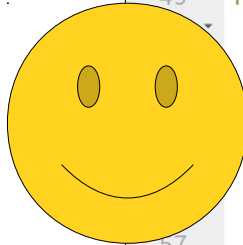
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

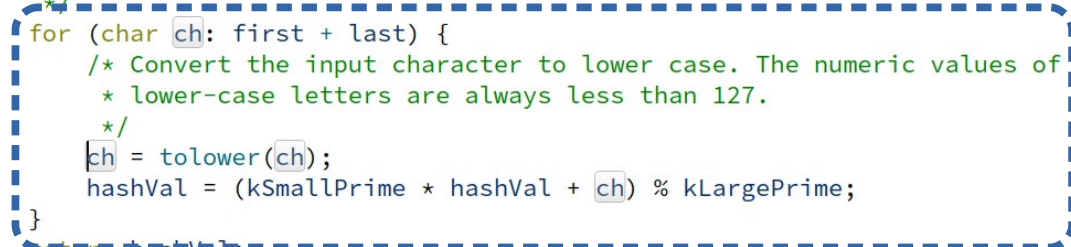
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

This loop is a **range-based for loop**. It says "for each character in the string first + last, do something with that character."



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```



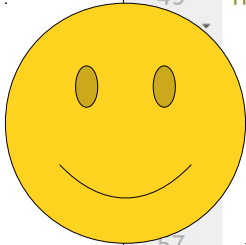
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace" 65
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Remember (from a while back) that we entered the name **Ada Lovelace**.



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

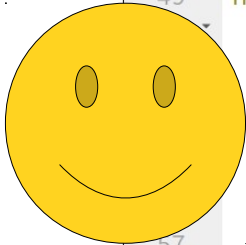
Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring) ...eHash.cpp	66	...	555892674			(all)

Name	Value	Type

If we take a look at the current value of the variable `ch`, we can see that it has the value `A`. That's the first letter of the name Ada Lovelace.



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

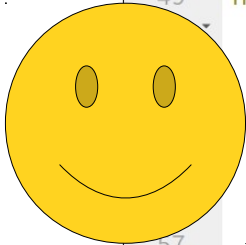
Name	Value
<code>__for_begin</code>	65 'A'
<code>__for_end</code>	0 '\000'
<code>__for_range</code>	"AdaLovelace"
<code>ch</code>	'A' 65
<code>first</code>	Ada
<code>hashVal</code>	0
<code>kLargePrime</code>	16908799
<code>kSmallPrime</code>	127
<code>last</code>	"Lovelace"

Debugger - GDB for "Welcome to CS106B" - Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

So now we know where we are (line 66), how we got there (main called nameHash), and the values in the program at this point.



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

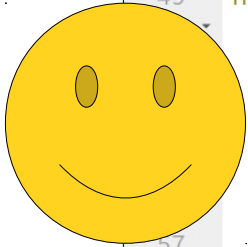
Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring) ...eHash.cpp	66	...	555892674			(all)

Now, let's do something really cool - we're going to run this program one line at a time, watching what happens at each step!



```
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger - GDB for "Welcome to CS106B" - Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

Qt Creator interface showing the project explorer on the left, the code editor in the center, the variable viewer on the right, and the debugger console at the bottom. The debugger is currently stopped at a breakpoint in the nameHash function.

Projects

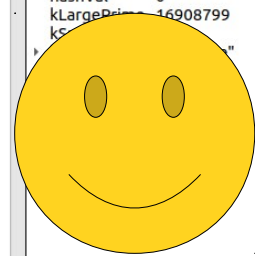
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the last
60  * name, updating the hash at each step.
61  */
62  for (ch = first; ch != '\0'; ch++)
63  {
64
65
66
67
68  }
69  return hashVal;
70  }
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127



Right above the stack trace, you'll see there are some small button icons.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

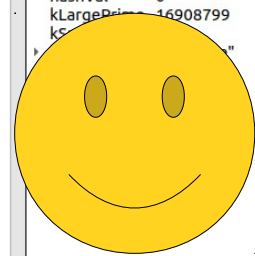
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...>::operator()() const	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64
65     for (ch : last)
66         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
67
68     return hashVal;
69 }
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127



These buttons let you resume the program, stop the program, walk through it one line at a time, etc.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

Debugger

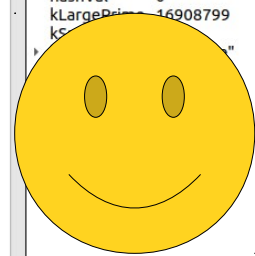
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (ch : first)
63         hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;
64
65     for (ch : last)
66         hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;
67
68     return hashVal;
69 }
70
71 }
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127



Move your mouse so that you're hovering over the button that's third from the left. If you hover over it, it should say "step over."

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
4	std::function<int (&)>::operator()() const	std_f...	706
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302
7	std::function<int (&)>::operator()() const	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring)	...eHash.cpp	66	...555892674			(all)

Qt

Debug

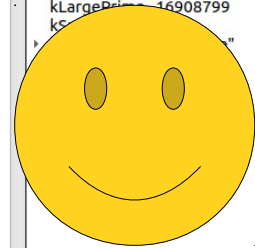
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (ch : first)
63         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
64
65     for (ch : last)
66         hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
67
68     return hashVal;
69 }
70
71 }
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127



Once you're confident that you're on the "Step Over" button - and not the "Step Into" or "Step Out" buttons - go and click it! When you do...



Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

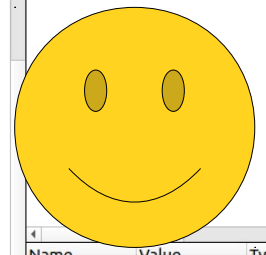
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the last
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovellace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovellace"



...your window should look something like this.

Debugger GDB for "Welc...

Level	Function
1	nameHash
2	qMain
3	std::Function_handle...
4	std::function<int (>::...
5	QtGui::<lambda>:::o...
6	std::Function_handle...
7	std::function<int (>::...
8	GThreadStd::run
9	GThreadStd::<lambda>:::operat...
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>>(std::invoke_o... invok... 60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the last
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



Okay! A few things have changed. Let's see what's going on.

Debugger GDB for "Wel...

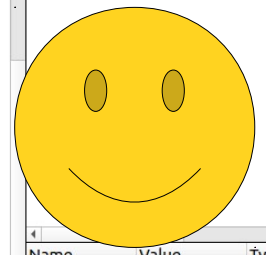
Level	Function
1	nameHash
2	qMain
3	std::Function_handl
4	std::function<int ()>::
5	QtGui::<lambda>::o
6	std::Function_handl
7	std::function<int ()>::
8	GThreadStd::run
9	GThreadStd::<lambda>::operat
10	std:: invoke_impl<void, GThreadStd::start()::<lambda>>>(std:: invoke_o... invok... 60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



First, notice that our helpful yellow Arrow friend is now pointing at line 67.

Debugger - GDB for "Welc...

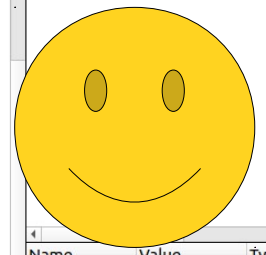
Level	Function
1	nameHash
2	qMain
3	std::Function_handl
4	std::function<int ()>::
5	QtGui::<lambda>:::o
6	std::Function_handl
7	std::function<int ()>::
8	GThreadStd::run
9	GThreadStd::<lambda>:::operat
10	std:: invoke_impl<void, GThreadStd::start():<lambda>>>(std:: invoke_o... invok... 60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



We're now at the line right after the one where we stopped. You just ran a single line of the program! Pretty cool!

Debugger GDB for "Wel...

Level	Function
1	nameHash
2	qMain
3	std::Function_handl...
4	std::function<int (>::...
5	QtGui::<lambda>:::o...
6	std::Function_handl...
7	std::function<int (>::...
8	GThreadStd::run
9	GThreadStd::<lambda>:::operat...
10	std:: invoke_impl<void, GThreadStd::start():<lambda>>>(std:: invoke_o... invok... 60

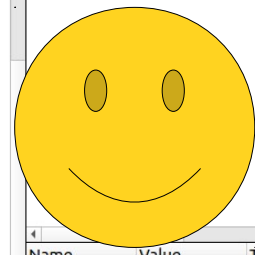
Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



so what did that line of code do?

Debugger GDB for "Welc...

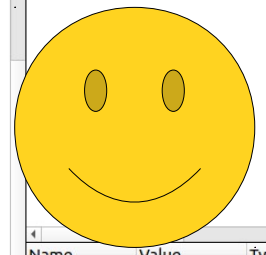
Level	Function
1	nameHash
2	qMain
3	std::Function_handle
4	std::function<int ()>::
5	QtGui::<lambda>>::o
6	std::Function_handle
7	std::function<int ()>::
8	GThreadStd::run
9	GThreadStd::<lambda>>::operator
10	std::invoke_impl<void, GThreadStd::start()>><lambda>>>(std::invoke_o... invok... 60

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



This line converts ch to lower case. The tolower function takes in a character and returns a lower-case version of it, so this overwrites ch with a lower-case version of itself.

Debugger GDB for "Wel...

Level	Function
1	nameHash
2	qMain
3	std::Function_handl
4	std::function<int ()>::o
5	QtGui::<lambda>>::o
6	std::Function_handl
7	std::function<int ()>::o
8	GThreadStd::run
9	GThreadStd::<lambda>>::operat
10	std:: invoke_impl<void, GThreadStd::start()>><lambda>>>std:: invoke_o... invok... 60

Projects

- Welcome to CS106B [master]
- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Qt

- Welcome
- Edit
- Debug
- Projects
- Help

Qt

You can actually see this by looking at the values panel over on the side!



```
55 int kLargePrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (>, int (>)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>>::operator()() const	std_f...	706								
5	QtGui::<lambda>>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (>>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Notice that the value associated with ch has changed from 'A' to 'a' - it's now in lower-case!



```
55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	AdaLovelace
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67								
2	qMain	Nam...	31	1	...ring)	...eHash.cpp	66	...555892674			(all)
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
- Welcome to CS106B.pro
- Headers
- Sources
- lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

If you'll notice, this value is in red while all the other values are in black.



```
55 int kLargePrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	AdaLovelace
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67								
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (&)*>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

This indicates that the value here has changed since the previous step. This is a really useful way to keep track of what's changing as you run the program.



```
55 int prime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	AddrLovelace
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67								
2	qMain	Nam...	31	1	...ring	...eHash.cpp	66	...555892674			(all)
3	std::Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start():<lambda>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Now, let's take a look at line 67, where we are right now.



```
55 int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (>, int (>):_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>):operator()() const	std_f...	706								
5	QtGui::lambda():operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (>):operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::lambda():operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start():lambda()>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files



Not gonna lie, this is a pretty dense line of code. It performs some weird sort of mathematical calculation on a bunch of different values.

```
55 int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67								
2	qMain	Nam...	31	1	...ring	...eHash.cpp	66	...555892674			(all)
3	std::Function_handler<int (>, int (>)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>::operator()() const	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::lambda()>>std::invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Fundamentally, though, it's just computing some weird function of some values and stashing it into hashVal.



```

55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (>, int (>)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>>::operator()() const	std_f...	706								
5	QtGui::<lambda(>>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int (>>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda(>>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda(>>(std::_invoke_o...	invok...	60								

Qt

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Projects

- Welcome to CS106B [master]
- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Let's go run that line of code and see what happens!



```
55 int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

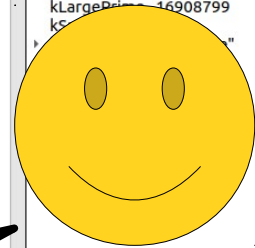
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67								
2	qMain	Nam...	31	1	...ring)	...eHash.cpp	66	...555892674			(all)
3	std::Function_handler<int (>, int (>)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (>::operator()() const	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (>, QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::lambda()>>std::invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the
60     * name, updating the hash at each step.
61     */
62
63
64
65
66
67
68
69
70
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127



Hover over the "step Over" button, confirm that the button you're clicking really is "step Over," and click it! When you do...



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads	
1	nameHash	Nam...	67	1	...	ring	...	eHash.cpp	66	...	555892674	(all)
2	qMain	Nam...	31									
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302									
4	std::function<int (&)>::operator()() const	std_f...	706									
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981									
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302									
7	std::function<int (&)>::operator()() const	std_f...	706									
8	GThreadStd::run	spl.cpp	22491									
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514									
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::invoke_o...	invok...	60									

Projects

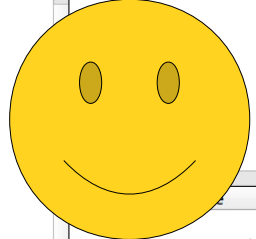
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the last
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63      /* Convert the input character to lower case. The numeric values of
64      * lower-case letters are always less than 127.
65      */
66      ch = tolower(ch);
67      hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



... you'll end up with something like this!

Debugger GDB for

Level	Function
1	nameHash
2	qMain
3	std::Function
4	std::function<
5	QtGui::lambda
6	std::Function
7	std::function<int (>::operator()
8	GThreadStd::run
9	GThreadStd::lambda>::operator()
10	std::invoke_impl<void, GThreadStd::start()>

Debugger

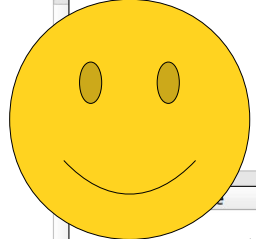
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

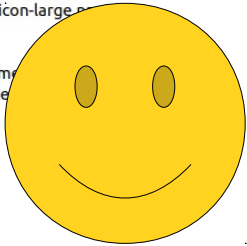


Let's see what's changed.

Debugger GDB for

Level	Function
1	nameHash
2	qMain
3	std::Function
4	std::function<
5	QtGui::<lambd
6	std::Function
7	std::function<int (>::operator()
8	GThreadStd::run
9	GThreadStd::<lambda>::operator()
10	std::invoke_impl<void, GThreadStd::start():<lambda>>>(std::invo...

First, notice that the value stored in hashVal changed to 97. We know that it changed because the value is in red, and we know that nothing else changed because nothing else is in red!



```
52
53
54 static const int kLargePrime = 16908799;
55 static const int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric values of
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start():<lambda()>>(std::invoke o...	invok...	60								

Projects

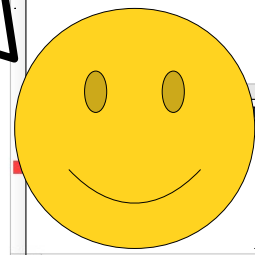
- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - k
53  */
54  stat
55  stat
56
57  int
58
59  /* I
60  * n
61  */
62  for (cha
63
64  /* Convert the input character to lower case. The num
65  * lower-case letters are always less than 127.
66  */
67  ch = tolower(ch);
68  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69
70  }
71

```

Second, notice that we're back up at the top of the for loop, since that's where the yellow arrow is pointing. We ended up back here because this is the next line that gets executed.



Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...>::operator()() const	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>::std::_invoke_o...	invok...	60								

Level Function File Line

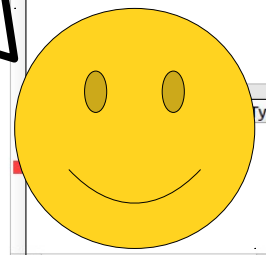
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the string.
60     * Note that we use the character's ASCII value as the hash value.
61     */
62     for (int i = 0; i < first.length(); i++)
63     {
64         /* Convert the input character to lower case. The number of
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(first[i]);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

We just single-stepped through a single iteration of that loop! Pretty cool!



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...(ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::lambda()>>std::invoke_o...	invok...	60								

Debugger

- Issues
- Search Results
- Application Output
- Compile Output
- QML Debugger Console
- General Messages
- Test Results

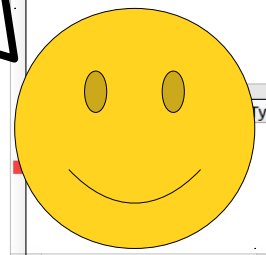
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters of the string.
60     * Note that we iterate over the characters of the string, not the
61     * characters of the string itself.
62     */
63     for (int i = 0; i < first.length(); i++)
64         /* Convert the input character to lower case. The number of
65         * lower-case letters are always less than 127.
66         */
66         ch = tolower(first[i]);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Let's go do it again!



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...(ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

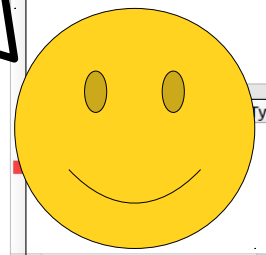
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the string.
60     * Note that we use the string's length to iterate over the characters.
61     */
62     for (int i = 0; i < first.length(); i++)
63         /* Convert the input character to lower case. The number of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(first[i]);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Again, move your mouse over the step over button (and make sure it says "step over" and not something else!), then click it.



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThreadStd::run)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>::std::_invoke_o...	invok...	60								

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

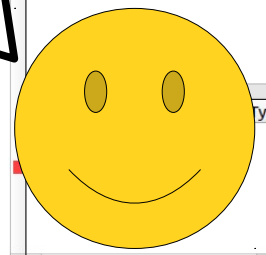
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the first string.
60     * Note that we use the character 'd' as a test case.
61     */
62     for (int i = 0; i < first.length(); i++){
63         /* Convert the input character to lower case. The number of
64         * lower-case letters are always less than 127.
65         */
66         char ch = tolower(first[i]);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	Ada Lovelace
ch	'd'
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Now we're here! Notice that ch now has the value 'd', which is the second letter of the name Ada.



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...(ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::lambda<>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::lambda<>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::lambda<>>::std::invoke_o...	invok...	60								

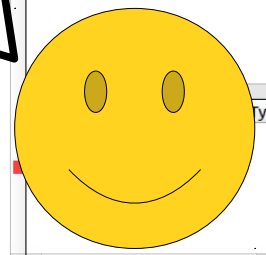
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the string.
60     * Note that we use the character at index 0, not the first character.
61     */
62     for (int i = 0; i < first.length(); i++) {
63         /* Convert the input character to lower case. The number of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(first[i]);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Go click "Step Over" again to run this line of code.



Debugger GDB for "Welcome to CS106B" Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

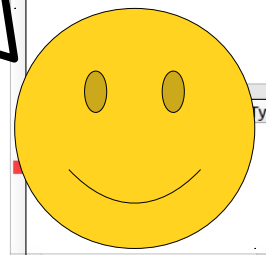
```

47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int
58
59     /* I
60     * n
61     */
62     for (c
63     /* Convert the input character to lower case. The num
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71

```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

You should be here now. Notice that none of the values changed. That makes sense, since all we did was convert a lower-case 'd' to a lower-case 'd'.



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...>	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::<lambda()>>std::invoke_o...	invok...	60								

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

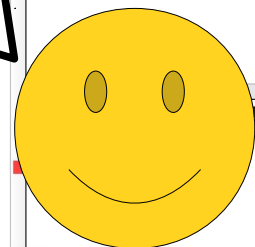
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int
58
59     /* I
60     * n
61     */
62     for (C
63     /* Convert the input character to lower case. The num
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Now, click "Step Over" one more time.



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Qt

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

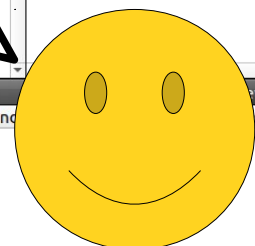
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

You'll now be at this point in the program. We've covered up the value of hashVal in this image, because at this point you should be able to see what hashVal is by reading the value in the side pane. This is the special value we want you to tell us when submitting the assignment!



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Cond
1	nameHash	Nam...	62						
2	qMain	Nam...	31	1	...(ring)	...eHash.cpp	66	...555892674	
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302						
4	std::function<int (&)>::operator()() const	std_f...	706						
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981						
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkn...	std_f...	302						
7	std::function<int (&)>::operator()() const	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514						
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60						

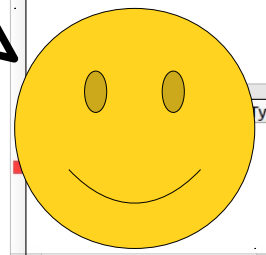
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
55
56
57
58
59
60
61
62
63     /* Convert the input character to lower case.
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	???
kLargePrime	10536799
kSmallPrime	127
last	"Lovelace"

To finish up this section on the debugger, we'd like to show you two last little techniques that you might find useful when debugging programs.



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674			(all)
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()>::<lambda>>(std::_invoke_o...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

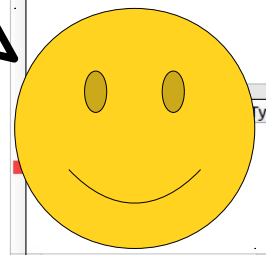
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
55     /* Convert the input character to lower case.
56     * lower-case letters are always less than 127.
57     */
58     ch = tolower(ch);
59     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
60 }
61 return hashVal;
62 }
63
64
65
66
67
68
69
70
71
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	???
kLargePrime	10536799
kSmallPrime	127
last	"Lovelace"

To start this off, click on the the breakpoint that we set earlier in the program. If you do...



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31	1	...(ring)	...eHash.cpp	66	...555892674			(all)
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invo...	invok...	60								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

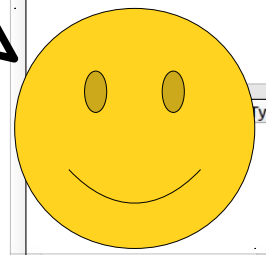
Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
55
56     /* Convert the input character to lower case.
57     * lower-case letters are always less than 127.
58     */
59     ch = tolower(ch);
60     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
61 }
62 return hashVal;
63 }
64
65
66
67
68
69
70
71
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	???
kLargePrime	1000000007
kSmallPrime	127
last	"Lovelace"

... it should clear the breakpoint. Now, if we were to run this program again in debug mode, it would not stop at this point, since nothing's telling it to!



Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkin...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda>>(std::invoke_o...	invok...	60								

Qt

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then
60     * name, updating the hash at each step.
61     */
62     for (int i = 0; i < first.length(); ++i)
63         hashVal = (hashVal * kLargePrime + first[i]) % kSmallPrime;
64
65     for (int i = 0; i < last.length(); ++i)
66         hashVal = (hashVal * kLargePrime + last[i]) % kSmallPrime;
67
68     }
69     return hashVal;
70 }
71 }
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



Now, take a look back at these buttons.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47  * but we thought it might be fun!)
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then
60  * name, updating the hash at each step.
61  */
62  for (int i = 0; i < first.length(); i++)
63  {
64
65
66
67
68  }
69  return hashVal;
70  }
71

```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



Hover your mouse over the one that's on the far right. When you hover over it, it should say "step out."



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThun...	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>::std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then
60     * name, updating the hash at each step.
61     */
62     for (int i = 0; i < first.length(); i++)
63         hashVal = (hashVal * kLargePrime + first[i]) % kSmallPrime;
64
65     for (int i = 0; i < last.length(); i++)
66         hashVal = (hashVal * kLargePrime + last[i]) % kSmallPrime;
67
68     }
69     return hashVal;
70 }
71 }
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



If you click this button, it will keep running this function up until it completes and returns.

Debugger GDB for "Welcome to CS106B" Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then
60     * name, updating the hash at each step.
61     */
62     for (int i = 0; i < first.length(); i++)
63         hashVal = (hashVal * kLargePrime + first[i]) % kSmallPrime;
64
65     for (int i = 0; i < last.length(); i++)
66         hashVal = (hashVal * kLargePrime + last[i]) % kSmallPrime;
67
68     }
69     return hashVal;
70 }
71 }
```

Name	Value
__for_begin	100 'd'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



Now, go click that button. If you did everything right...

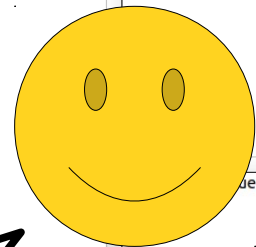
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::Function_handler<int (&), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int (&)>::operator()() const	std_f...	706								
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
6	std::Function_handler<int (&), QtGui::startBackgroundEventLoop(GThinkIn...)	std_f...	302								
7	std::function<int (&)>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code
38  * to talk more
39  * the m
40  * of th
41  *
42  * For t
43  * treat
44  * It th
```

Name	Value	Type
first	"Ada"	std::st
hashValue	21845	int
last	"Lovelace"	std::st



... you should end up with something that looks like this!

Debugger GDB for "Wel

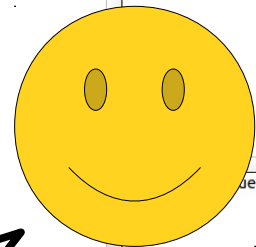
Level	Function
1	qMain
2	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)
3	std::function<int ()>::operator()() const
4	QtGui::<lambda()>::operator()(void) const
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...>::_M_invoke(std::_Any_data const&)
6	std::function<int ()>::operator()() const
7	GThreadStd::run
8	GThreadStd::<lambda()>::operator()(void) const
9	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...
10	std::_invoke<GThreadStd::start()::<lambda()>>(GThreadStd::<lambda()> &&) invok...

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code
38  * to talk more
39  * the m
40  * of th
41  *
42  * For t
43  * treat
44  * It th
```

Name	Value	Type
first	"Ada"	std::st
hashValue	21845	int
last	"Lovelace"	std::st



Let's take a minute to get our bearings.
Where exactly are we?

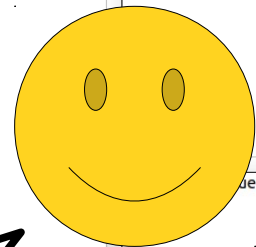
Debugger - GDB for "Wel...

Level	Function
1	qMain
2	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)
3	std::function<int ()>::operator()() const
4	QtGui::<lambda()>::operator()(void) const
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...>::_M_invoke(std::_Any_data const&)
6	std::function<int ()>::operator()() const
7	GThreadStd::run
8	GThreadStd::<lambda()>::operator()(void) const
9	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...
10	std::_invoke<GThreadStd::start()::<lambda()>>(GThreadStd::<lambda()> &&) invok...

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code
38 * to talk more
39 * the m
40 * of th
41 *
42 * For t
43 * treat
44 * It th
```



Well, the yellow arrow indicates that we're back in main again. Cool!

Name	Value	Type
first	"Ada"	std::st
hashValue	21845	int
last	"Lovelace"	std::st

Name	Value	Type
de	1967457	int

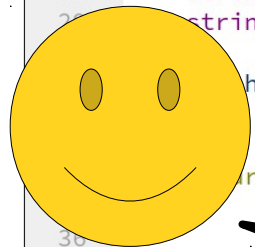
Debugger GDB for "Wel

Level	Function
1	qMain
2	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)
3	std::function<int ()>::operator()() const
4	QtGui::<lambda()>::operator()(void) const
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...>::_M_invoke(std::_Any_data const&)
6	std::function<int ()>::operator()() const
7	GThreadStd::run
8	GThreadStd::<lambda()>::operator()(void) const
9	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...
10	std::_invoke<GThreadStd::start()::<lambda()>>(GThreadStd::<lambda()> &&) invok...

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the function that computes the hash code. We're going
38 * to talk more about it later.
39 * the main function calls this function.
40 * of the hash code.
41 *
42 * For the hash code, we're going to use a simple algorithm.
43 * treat each character as a number from 0 to 255.
44 * It then computes the hash code by adding up the squares of the numbers.
```



We can see that the nameHash function returned 1967457. Thanks, debugger!

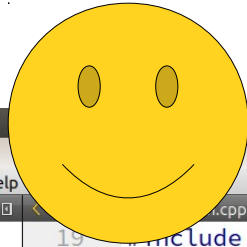
Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

returned value 1967457

Name	Value	Type
------	-------	------

Debugger - GDB for "Welcome to CS106B"

Level	Function
1	qMain
2	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)
3	std::function<int ()>::operator()() const
4	QtGui::<lambda()>::operator()(void) const
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...>::_M_invoke(std::_Any_data const&)
6	std::function<int ()>::operator()() const
7	GThreadStd::run
8	GThreadStd::<lambda()>::operator()(void) const
9	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...
10	std::_invoke<GThreadStd::start()::<lambda()>>(GThreadStd::<lambda()> &&)



NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

<Select Symbol> Unix (LF) Line: 31, Col: 5

```
19 #include "simpleHash.h" // for getLine
20 using namespace std;
21
22 /* Prototype
23 * in main a
24 */
25 int nameHash(const string& name)
26
27 int main()
28 string name;
29 string line;
30
31 int hashValue = 0;
32
33 cout << "The hash of your name is: " << hashValue << endl;
34 return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```

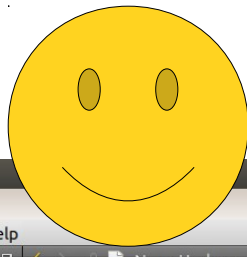
But if you look up over here in the values window, you can see that hashValue has some really weird-looking number stored in it. (You'll almost certainly see something different on your system.)

Name	Value	Type
hashValue	21845	int

Name	Value	Type
returned value	1967457	int

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	31								
2	std::Function_handler<int (), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
5	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...)	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::lambda()::operator()(void) const	spl.cpp	22514								
9	std::_invoke_impl<void, GThreadStd::start()::lambda()>>(std::_invoke_o...	invok...	60								
10	std::_invoke<GThreadStd::start()::lambda()>>(GThreadStd::lambda() &&)	invok...	95								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results



But it looks like we're setting hashValue equal to the number that was returned by the nameHash function. What's going on?

Qt Creator interface showing a C++ project named "Welcome to CS106B". The main editor displays the file "NameHash.cpp" with the following code:

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype
23 * in main a
24 */
25 int nameHash(const string& first, const string& last);
26
27 int main()
28 {
29     string first;
30     string last;
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```

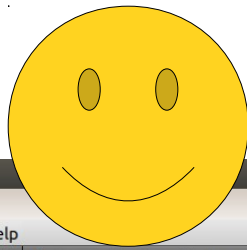
The variable `hashValue` is highlighted in blue in the code. The right-hand side of the interface shows a variable inspector with the following data:

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

Below the code editor is a debugger window showing the call stack:

Level	Function	File	Line
1	qMain	Nam...	31
2	std::_Function_handler<int (int (*)>::M_invoke(std::_Any_data const&)	std_f...	302
3	std::function<int (int (*)>::operator()() const	std_f...	706
4	QtGui::lambda()::operator()(void) const	spl.cpp	20981
5	std::_Function_handler<int (int (*)>::M_invoke(std::_Any_data const&)	std_f...	302
6	std::function<int (int (*)>::operator()() const	std_f...	706
7	GThreadStd::run	spl.cpp	22491
8	GThreadStd::lambda()::operator()(void) const	spl.cpp	22514
9	std::_invoke_impl<void, GThreadStd::start()::lambda()>(std::_invoke_o...	invok...	60
10	std::_invoke<GThreadStd::start()::lambda()>(GThreadStd::lambda() &&)	invok...	95

The status bar at the bottom indicates the application is stopped at "function-finished".



This is pretty cool, actually!

Qt Creator interface showing a C++ project named "Welcome to CS106B". The main editor displays the source code for "NameHash.cpp".

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype
23 * in main a
24 */
25 int nameHash(const string& first, const string& last);
26
27 int main()
28 {
29     string first;
30     string last = getLine("What is your last name? ");
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```

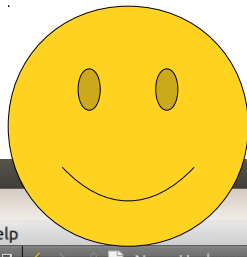
The right-hand side of the interface shows a variable inspector window with the following data:

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
	"Lovelace"	std::string

Below the code editor is a debugger window showing the call stack:

Level	Function	File	Line
1	qMain	Nam...	31
2	std::_Function_handler<int (int (*))>::M_invoke(std::_Any_data const&)	std_f...	302
3	std::function<int (int)>::operator()(int) const	std_f...	706
4	QtGui::lambda()::operator()(void) const	spl.cpp	20981
5	std::_Function_handler<int (int)>::M_invoke(std::_Any_data const&)	std_f...	302
6	std::function<int (int)>::operator()(int) const	std_f...	706
7	GThreadStd::run	spl.cpp	22491
8	GThreadStd::lambda()::operator()(void) const	spl.cpp	22514
9	std::_invoke_impl<void, GThreadStd::start()::lambda()>>::M_invoke_0...	invok...	60
10	std::_invoke<GThreadStd::start()::lambda()>>(GThreadStd::lambda() &&)	invok...	95

The bottom status bar shows various toolbars and tabs for debugging, including "Issues", "Search Results", "Application Output", "Compile Output", "QML Debugger Console", "General Messages", and "Test Results".



NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

What's happened is that we've just returned from nameHash with a value, but since we're going through the program one step at a time, we haven't actually assigned that value to hashValue yet!

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype
23 * in main
24 */
25 int nameHash(const string& s, int n);
26
27 int main()
28 {
29     string first;
30     string last;
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
	"Lovelace"	std::string

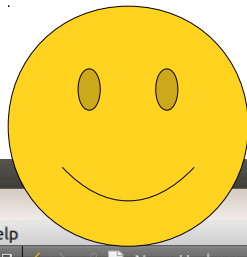
returned value 1967457 int

Name	Value	Type
------	-------	------

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	31								
2	std::_Function_handler<int (), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThrunkin...)	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
9	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								
10	std::_invoke<GThreadStd::start()::<lambda()>>(GThreadStd::<lambda()> &&)	invok...	95								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results



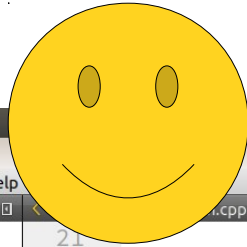
Let's do a "step over" so that we can finish executing this line. Click "step over," and if you did everything right...

Qt Creator interface showing a C++ project named "Welcome to CS106B". The main editor displays the source code for "NameHash.cpp".

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype
23 * in main
24 */
25 int nameHash(const string& s1, const string& s2);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
```

The debugger window at the bottom shows the call stack. The "Step Over" button (represented by a red square) is highlighted. The stack includes:

Level	Function	File	Line
1	qMain	Nam...	31
2	std::Function_handler<int (), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302
3	std::function<int ()>::operator()() const	std_f...	706
4	QtGui::lambda()::operator()(void) const	spl.cpp	20981
5	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302
6	std::function<int ()>::operator()() const	std_f...	706
7	GThreadStd::run	spl.cpp	22491
8	GThreadStd::lambda()::operator()(void) const	spl.cpp	22514
9	std::_invoke_impl<void, GThreadStd::start()::lambda()>>(std::_invoke_o...	invok...	60
10	std::_invoke<GThreadStd::start()::lambda()>>(GThreadStd::lambda() &&)	invok...	95



... you should see the right value get stored (notice it's in red!) and we've moved to the next line.

Qt Creator interface showing a C++ project named "Welcome to CS106B [master]". The main editor displays the source code for "NameHash.cpp".

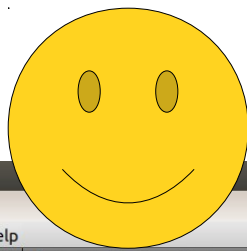
```
21
22 /* Prototype for the function. This lets us use the function
23 * in main and the
24 */
25 int nameHash(const string& first, const string& last)
26
27 int main()
28     string first, last;
29     string input;
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS106B,
```

The right-hand pane shows a variable watch table:

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int
last	"Lovelace"	std::string

The bottom pane shows the GDB debugger stack trace:

Level	Function	File	Line
1	qMain	Nam...	33
2	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302
3	std::function<int ()>::operator()(void) const	std_f...	706
4	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...)	std_f...	302
6	std::function<int ()>::operator()(void) const	std_f...	706
7	GThreadStd::run	spl.cpp	22491
8	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
9	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60
10	std::_invoke<GThreadStd::start()>::<lambda()>>(GThreadStd::<lambda()> &&)	invok...	95



NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

At this point, we've seen just about everything we care about. Rather than single-stepping all the way to the end, let's just tell the program to keep on running.

```
21
22 /* Prototype
23 * in main
24 */
25 int nameHash
26
27 int main()
28     string
29     string
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS106B,
```

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int
last	"Lovelace"	std::string

Debugger GDB for "Welcome to CS106B" Stopped: "end-stepping-range"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	33								
2	std::_Function_handler<int (), int (*)>::_M_invoke(std::_Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()(void) const	std_f...	706								
4	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...)	std_f...	302								
6	std::function<int ()>::operator()(void) const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
9	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								
10	std::_invoke<GThreadStd::start()>::<lambda()>>(GThreadStd::<lambda()> &&)	invok...	95								

Name	Value	Type
------	-------	------

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash. We're going
38  * to talk more about what hash functions do later in the course. In
39  * the meantime, think of it as a function that scratches the characters
40  * of the input and produces a number.
41  *
42  * For those of you who are mathematically inclined, this function

```

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int
last	"Lovelace"	std::string



To do this, click on this button. If you hover over it, it says "Continue," and that button means "unpause the program and let it keep running from here."

Qt Creator interface showing a yellow box around the 'Continue' button in the debug console area.

```

8 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22491
9 std::__invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::__invoke_o... spl.cpp 22514
10 std::__invoke<GThreadStd::start()>::<lambda()>>(GThreadStd::<lambda()> &&) invok... 60

```

Number	Function	File	Line	Address	Condition	Ignore	Threads
Stopped: "end-stepping-range".							



If you do, you should see something like this.
(The program window might not automatically pop up. That's okay! Just open it manually.)
Our program is now done running!

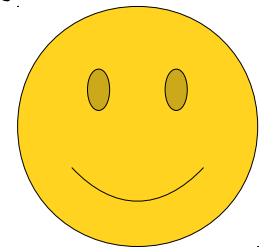
```
27 int main()
28 {
29     QString str1, str2;
30     int i;
31     int j;
32     int k;
33     cout << endl;
34     return 0;
35 }
36
37 /* This
38 * to ta
39 * the m
40 * of th
41 *
42 * For t
43 * treat
44 * It th
45 * F_p,
46 * some
```

```
File Edit Options Help
What is your first name? Ada
What is your last name? Lovelace
The hash of your name is: 1967457
```

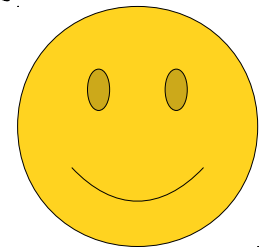
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	33								
2	std::_Function_handler<int (), int (*)>::M_invoke(std::_Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981								
5	std::_Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...)	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
9	std::_invoke_impl<void, GThreadStd::start()>::<lambda()>>(std::_invoke_o...	invok...	60								
10	std::_invoke<GThreadStd::start()>::<lambda()>>(GThreadStd::<lambda()> &&)	invok...	95								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

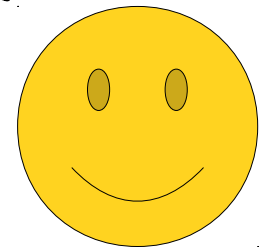
so there you have it! You've now gotten more familiar with the debugger!



You know how to set a breakpoint to pause the program at a particular point.



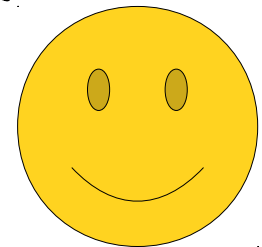
You know how to read the call stack and to see the values of local variables.



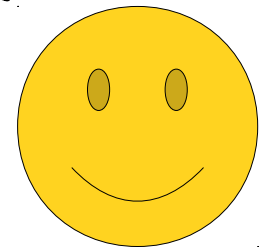
You know how to single-step the program and see what values change.



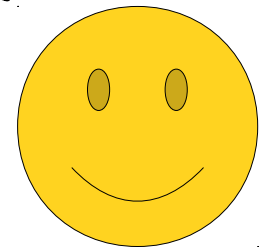
You know how to run a function to completion,
and how to let the program keep on running.



As you write more and more complicated programs this quarter, you'll get a lot more familiar using the debugger and seeing how your programs work.



And, if you continue to build larger and larger pieces of software, you'll find that knowing how to use a debugger is a surprisingly valuable skill!



Hope this helps, and welcome to CS106B!

