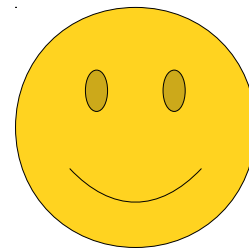
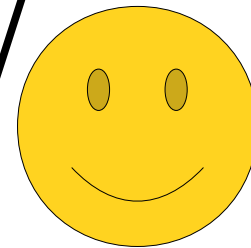


Assignment 0: Using the Debugger

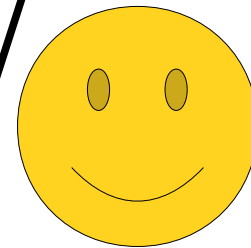
Hi everybody!



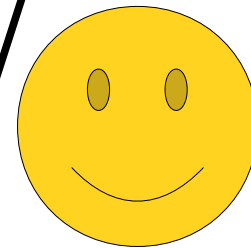
As part of Assignment 0, we'd like you to get a little bit of practice using the debugger in Qt Creator.



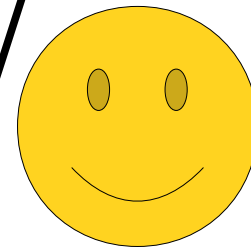
The debugger is a tool you can use to help see what your program is doing as you run it.



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.



```
40 * of the
41 *
42 * For the
43 * treats
44 * It then
45 * F_p, when
46 * some smaller prime numbers were chosen because their product is close to 2^31,
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
51     * prime. These numbers were chosen because their product is close to 2^31.
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
```

Qt Creator IDE interface showing the 'name_hash.cpp' file. The interface includes a top menu bar (File, Edit, Build, Debug, Analyze, Tools, Window, Help), a left sidebar with tool icons (Welcome, Edit, Design, Debug, Projects, Analyze, Help), a central code editor with the C++ code, and a bottom status bar with tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages. The system tray in the top right shows the time as 10:24 AM.

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```
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 * Fp where p is a large prime number, and evaluates that polynomial at  
this for CS106B,
```

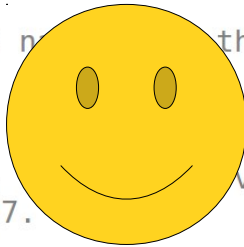
Move your mouse cursor so that it's in the space right before the line number for line 66.

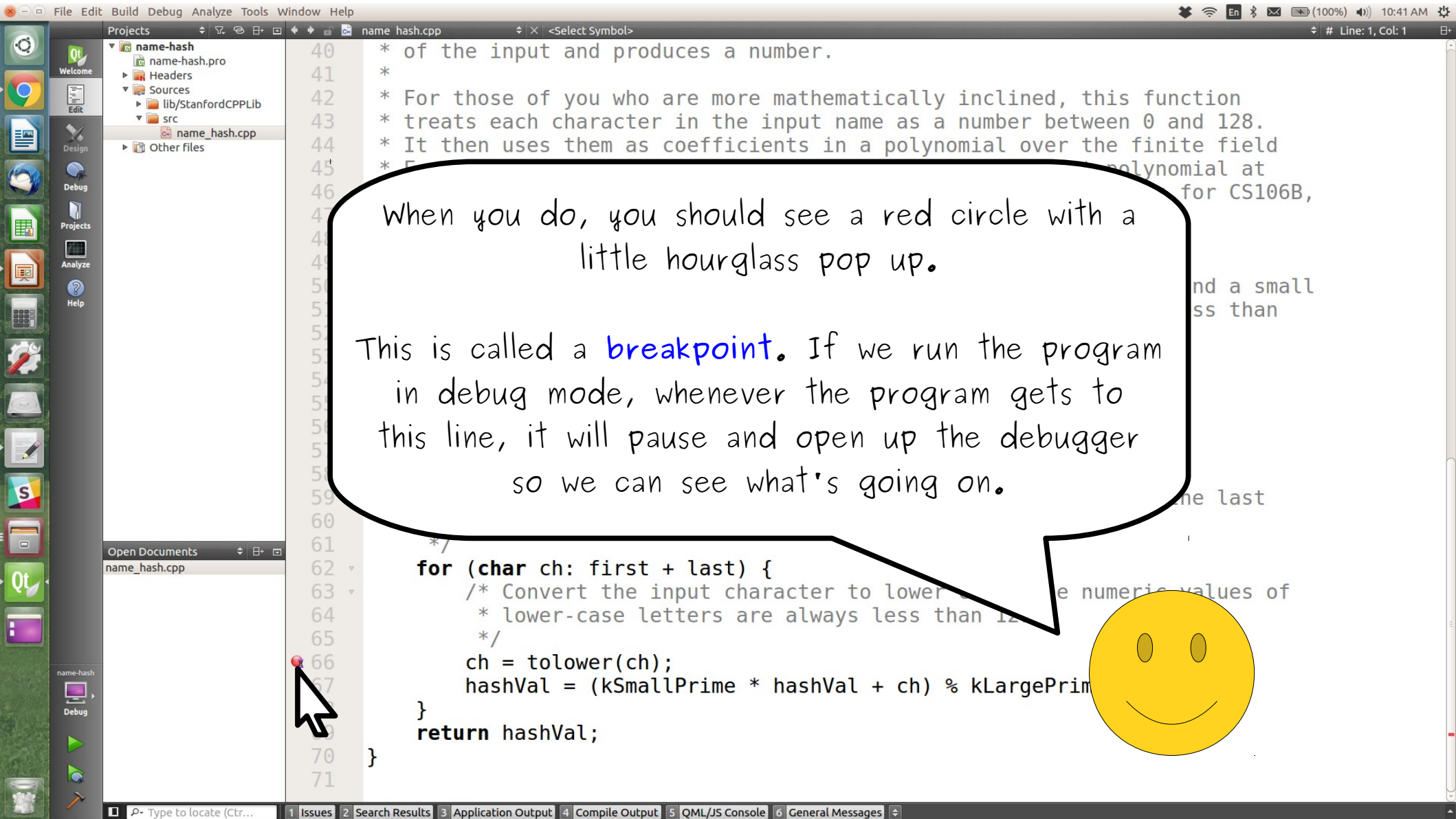
Now, click the mouse!

Open Documents

- name_hash.cpp

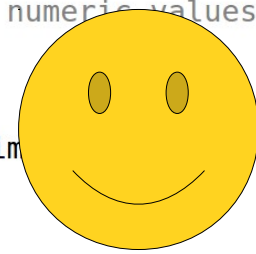
```
56  
57 int hashVal = 0;  
58  
59 /* Iterate across all the characters in the name, from the first to the last  
60 * name, updating the hash at each step.  
61 */  
62 for (char ch: first + last) {  
63     /* Convert the input character to lower case.  
64     * lower-case letters are always less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





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

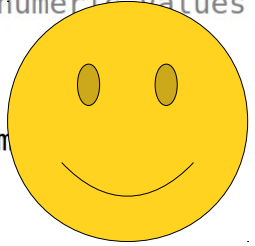
This is called a **breakpoint**. If we run the program in debug mode, whenever the program gets to this line, it will pause and open up the debugger so we can see what's going on.



Qt IDE interface showing a project tree on the left with 'name_hash.cpp' selected. The bottom-left corner features a vertical toolbar with icons for Run, Debug, and other actions. A mouse cursor is pointing at the Debug icon.

```
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 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51        prime, both less than
52        the last
53        the numeric values of
54        lower-case letters are always less than 12
55        */
56     for (char ch: first + last) {
57         ch = tolower(ch);
58         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
59     }
60     return hashVal;
61 }
```

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.



- 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 a C++ file named `name_hash.cpp` with the following code:

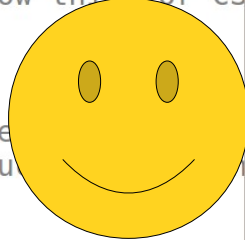
```
41
42
43 * treats each character in the input name as a coefficient in a polynomial
44 * It then uses them as coefficients in a polynomial of the form
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!)
48 */
49 int nameHash(const QString &name)
50 {
51     /*
52      *
53      *
54      *
55      *
56      *
57      *
58      *
59      */
60     int hashVal = 0;
61     for (int i = 0; i < name.length(); ++i)
62     {
63         char ch = name[i].toLatin1();
64         /*
65          *
66          *
67          */
68         ch = tolower(ch);
69         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
70     }
71     return hashVal;
72 }
```

A console window titled "Console" is open, displaying the prompt "What is your first name? |".

The debugger window at the bottom shows the following table:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

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



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



Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays code for a hash function. A console window is open, showing the program's output: 'What is your first name? Ada' and 'What is your last name?'. The console window also shows the Qt logo and menu options (File, Edit, Options, Help). The bottom status bar shows 'Application started' and a table of threads.

```
41 int nameHash(const char* name) {
42     int hashVal = 0;
43     * treats each character in the input name as a coefficient in a polynomial
44     * It then uses them as coefficients in a polynomial of degree n-1, where n is the length of the name.
45     * F_p, where p is a large prime number, and evaluates that polynomial at a
46     * some smaller prime number q. (You aren't expected to know this for CS
47     * but we thought it might be fun!)
48 */
49 int nameHash(const char* name) {
50     /*
51     /*
52     /*
53     /*
54     static const int kSmallPrime = 101;
55     static const int kLargePrime = 1000000007;
56
57     int n = strlen(name);
58
59     /*
60     /*
61     /*
62     for (int i = 0; i < n; i++) {
63         int ch = name[i];
64
65         /*
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69
70     return hashVal;
71 }
```

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

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



Qt IDE interface showing a C++ project named "name-hash". The main editor displays code for a function that calculates a hash value based on input names. A console window is open, showing the program's output: "What is your first name? Ada" and "What is your last name? Lovelace". The console window is partially overlapping the main editor. The Qt IDE interface includes a sidebar with various tool icons, a top menu bar, and a bottom status bar.

```
41 int nameHash(const QString &name) {
42     // This function
43     * treats each character in the input name as a coefficient in a polynomial
44     * It then uses them as coefficients in a polynomial of degree n-1 for the finite field
45     * F_p, where p is a large prime number, and evaluates that polynomial at a
46     * some smaller prime number q. (You aren't expected to know this for CS
47     * but we thought it might be fun!)
48     */
49     int nameHash(const QString &name) {
50     /*
51     * What is your first name? Ada
52     * What is your last name? Lovelace
53     *
54     *
55     *
56     *
57     *
58     *
59     */
60     *
61     *
62     *
63     *
64     *
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
```

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(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 interface. The main editor displays a C++ file named `name_hash.cpp` with the following code:

```
41 int nameHash(const QString &name) {
42     /*
43      * treats each character in the input name as a coefficient in a polynomial
44      * It then uses them as coefficients in a polynomial of degree n-1 for the finite field
45      * F_p, where p is a large prime number, and evaluates that polynomial at a
46      * some smaller prime number q. (You aren't expected to know this for CS
47      * but we thought it might be fun!)
48      */
49     int hashVal = 0;
50     for (int i = 0; i < name.length(); ++i) {
51         /*
52          * What is your first name? Ada
53          * What is your last name? Lovelace
54          */
55         char ch = name[i].toLatin1();
56         int chVal = ch - 'a';
57         hashVal = (kSmallPrime * hashVal + chVal) % kLargePrime;
58     }
59     return hashVal;
60 }
61
62 #include <QString>
63 #include <QStringList>
64 #include <QDebug>
65
66 int main() {
67     QString name = "Ada Lovelace";
68     int hash = nameHash(name);
69     qDebug() << "Name: " << name << " Hash: " << hash;
70 }
```

The console window shows the program's execution output:

```
Application started
What is your first name? Ada
What is your last name? Lovelace
```

A yellow smiley face emoji is overlaid on the right side of the console window. The bottom status bar shows the following tabs: 1 Issues, 2 Search Results, 3 Application Output, 4 Compile Output, 5 QML/JS Console, 6 General Messages.

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

The screenshot shows an IDE with a project named 'name-hash'. The source file 'name_hash.cpp' contains the following code:

```
41 int nameHash(const std::string& name) {  
42     // This function  
43     * treats each character in the input name as a coefficient in a polynomial  
44     * It then uses them as coefficients in a polynomial for 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 CS  
47     * but we thought it might be fun!)  
48     */  
49     int hashVal = 0;  
50     for (int i = 0; i < name.length(); i++) {  
51         char ch = name[i];  
52         // Convert character to a numeric value between 0 and 128  
53         // by subtracting 'a' (or 'A' for uppercase letters).  
54         int chVal = ch - 'a';  
55         // If uppercase letter, subtract 26 more.  
56         if (ch < 'A') chVal -= 26;  
57         // Update hash value.  
58         hashVal = (kSmallPrime * hashVal + chVal) % kLargePrime;  
59     }  
60     return hashVal;  
61 }  
62  
63  
64  
65  
66 ch = tolower(ch);  
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
```

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 bottom of the IDE shows the 'Threads' panel with the following data:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

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

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



```
47  */
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
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
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  }
```

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

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



```
47  */
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
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
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  }
```

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51      * prime. These numbers were chosen because their product is less th
52      * 202714367. Primes: 1000000007, 1000000009, 1000000003, 1000000001,
53      * 1000000005, 1000000002, 1000000004, 1000000006, 1000000008, 1000000000,
54      * 1000000001, 1000000002, 1000000003, 1000000004, 1000000005, 1000000006,
55      * 1000000007, 1000000008, 1000000009, 1000000010, 1000000011, 1000000012,
56      * 1000000013, 1000000014, 1000000015, 1000000016, 1000000017, 1000000018,
57      * 1000000019, 1000000020, 1000000021, 1000000022, 1000000023, 1000000024,
58      * 1000000025, 1000000026, 1000000027, 1000000028, 1000000029, 1000000030,
59      * 1000000031, 1000000032, 1000000033, 1000000034, 1000000035, 1000000036,
60      * 1000000037, 1000000038, 1000000039, 1000000040, 1000000041, 1000000042,
61      * 1000000043, 1000000044, 1000000045, 1000000046, 1000000047, 1000000048,
62      * 1000000049, 1000000050, 1000000051, 1000000052, 1000000053, 1000000054,
63      * 1000000055, 1000000056, 1000000057, 1000000058, 1000000059, 1000000060,
64      * 1000000061, 1000000062, 1000000063, 1000000064, 1000000065, 1000000066,
65      * 1000000067, 1000000068, 1000000069, 1000000070, 1000000071, 1000000072,
66      * 1000000073, 1000000074, 1000000075, 1000000076, 1000000077, 1000000078,
67      * 1000000079, 1000000080, 1000000081, 1000000082, 1000000083, 1000000084,
68      * 1000000085, 1000000086, 1000000087, 1000000088, 1000000089, 1000000090,
69      * 1000000091, 1000000092, 1000000093, 1000000094, 1000000095, 1000000096,
70      * 1000000097, 1000000098, 1000000099, 1000000100, 1000000101, 1000000102,
71      * 1000000103, 1000000104, 1000000105, 1000000106, 1000000107, 1000000108,
72      * 1000000109, 1000000110, 1000000111, 1000000112, 1000000113, 1000000114,
73      * 1000000115, 1000000116, 1000000117, 1000000118, 1000000119, 1000000120,
74      * 1000000121, 1000000122, 1000000123, 1000000124, 1000000125, 1000000126,
75      * 1000000127, 1000000128, 1000000129, 1000000130, 1000000131, 1000000132,
76      * 1000000133, 1000000134, 1000000135, 1000000136, 1000000137, 1000000138,
77      * 1000000139, 1000000140, 1000000141, 1000000142, 1000000143, 1000000144,
78      * 1000000145, 1000000146, 1000000147, 1000000148, 1000000149, 1000000150,
79      * 1000000151, 1000000152, 1000000153, 1000000154, 1000000155, 1000000156,
80      * 1000000157, 1000000158, 1000000159, 1000000160, 1000000161, 1000000162,
81      * 1000000163, 1000000164, 1000000165, 1000000166, 1000000167, 1000000168,
82      * 1000000169, 1000000170, 1000000171, 1000000172, 1000000173, 1000000174,
83      * 1000000175, 1000000176, 1000000177, 1000000178, 1000000179, 1000000180,
84      * 1000000181, 1000000182, 1000000183, 1000000184, 1000000185, 1000000186,
85      * 1000000187, 1000000188, 1000000189, 1000000190, 1000000191, 1000000192,
86      * 1000000193, 1000000194, 1000000195, 1000000196, 1000000197, 1000000198,
87      * 1000000199, 1000000200, 1000000201, 1000000202, 1000000203, 1000000204,
88      * 1000000205, 1000000206, 1000000207, 1000000208, 1000000209, 1000000210,
89      * 1000000211, 1000000212, 1000000213, 1000000214, 1000000215, 1000000216,
90      * 1000000217, 1000000218, 1000000219, 1000000220, 1000000221, 1000000222,
91      * 1000000223, 1000000224, 1000000225, 1000000226, 1000000227, 1000000228,
92      * 1000000229, 1000000230, 1000000231, 1000000232, 1000000233, 1000000234,
93      * 1000000235, 1000000236, 1000000237, 1000000238, 1000000239, 1000000240,
94      * 1000000241, 1000000242, 1000000243, 1000000244, 1000000245, 1000000246,
95      * 1000000247, 1000000248, 1000000249, 1000000250, 1000000251, 1000000252,
96      * 1000000253, 1000000254, 1000000255, 1000000256, 1000000257, 1000000258,
97      * 1000000259, 1000000260, 1000000261, 1000000262, 1000000263, 1000000264,
98      * 1000000265, 1000000266, 1000000267, 1000000268, 1000000269, 1000000270,
99      * 1000000271, 1000000272, 1000000273, 1000000274, 1000000275, 1000000276,
100     * 1000000277, 1000000278, 1000000279, 1000000280, 1000000281, 1000000282,
101     * 1000000283, 1000000284, 1000000285, 1000000286, 1000000287, 1000000288,
102     * 1000000289, 1000000290, 1000000291, 1000000292, 1000000293, 1000000294,
103     * 1000000295, 1000000296, 1000000297, 1000000298, 1000000299, 1000000300,
104     * 1000000301, 1000000302, 1000000303, 1000000304, 1000000305, 1000000306,
105     * 1000000307, 1000000308, 1000000309, 1000000310, 1000000311, 1000000312,
106     * 1000000313, 1000000314, 1000000315, 1000000316, 1000000317, 1000000318,
107     * 1000000319, 1000000320, 1000000321, 1000000322, 1000000323, 1000000324,
108     * 1000000325, 1000000326, 1000000327, 1000000328, 1000000329, 1000000330,
109     * 1000000331, 1000000332, 1000000333, 1000000334, 1000000335, 1000000336,
110     * 1000000337, 1000000338, 1000000339, 1000000340, 1000000341, 1000000342,
111     * 1000000343, 1000000344, 1000000345, 1000000346, 1000000347, 1000000348,
112     * 1000000349, 1000000350, 1000000351, 1000000352, 1000000353, 1000000354,
113     * 1000000355, 1000000356, 1000000357, 1000000358, 1000000359, 1000000360,
114     * 1000000361, 1000000362, 1000000363, 1000000364, 1000000365, 1000000366,
115     * 1000000367, 1000000368, 1000000369, 1000000370, 1000000371, 1000000372,
116     * 1000000373, 1000000374, 1000000375, 1000000376, 1000000377, 1000000378,
117     * 1000000379, 1000000380, 1000000381, 1000000382, 1000000383, 1000000384,
118     * 1000000385, 1000000386, 1000000387, 1000000388, 1000000389, 1000000390,
119     * 1000000391, 1000000392, 1000000393, 1000000394, 1000000395, 1000000396,
120     * 1000000397, 1000000398, 1000000399, 1000000400, 1000000401, 1000000402,
121     * 1000000403, 1000000404, 1000000405, 1000000406, 1000000407, 1000000408,
122     * 1000000409, 1000000410, 1000000411, 1000000412, 1000000413, 1000000414,
123     * 1000000415, 1000000416, 1000000417, 1000000418, 1000000419, 1000000420,
124     * 1000000421, 1000000422, 1000000423, 1000000424, 1000000425, 1000000426,
125     * 1000000427, 1000000428, 1000000429, 1000000430, 1000000431, 1000000432,
126     * 1000000433, 1000000434, 1000000435, 1000000436, 1000000437, 1000000438,
127     * 1000000439, 1000000440, 1000000441, 1000000442, 1000000443, 1000000444,
128     * 1000000445, 1000000446, 1000000447, 1000000448, 1000000449, 1000000450,
129     * 1000000451, 1000000452, 1000000453, 1000000454, 1000000455, 1000000456,
130     * 1000000457, 1000000458, 1000000459, 1000000460, 1000000461, 1000000462,
131     * 1000000463, 1000000464, 1000000465, 1000000466, 1000000467, 1000000468,
132     * 1000000469, 1000000470, 1000000471, 1000000472, 1000000473, 1000000474,
133     * 1000000475, 1000000476, 1000000477, 1000000478, 1000000479, 1000000480,
134     * 1000000481, 1000000482, 1000000483, 1000000484, 1000000485, 1000000486,
135     * 1000000487, 1000000488, 1000000489, 1000000490, 1000000491, 1000000492,
136     * 1000000493, 1000000494, 1000000495, 1000000496, 1000000497, 1000000498,
137     * 1000000499, 1000000500, 1000000501, 1000000502, 1000000503, 1000000504,
138     * 1000000505, 1000000506, 1000000507, 1000000508, 1000000509, 1000000510,
139     * 1000000511, 1000000512, 1000000513, 1000000514, 1000000515, 1000000516,
140     * 1000000517, 1000000518, 1000000519, 1000000520, 1000000521, 1000000522,
141     * 1000000523, 1000000524, 1000000525, 1000000526, 1000000527, 1000000528,
142     * 1000000529, 1000000530, 1000000531, 1000000532, 1000000533, 1000000534,
143     * 1000000535, 1000000536, 1000000537, 1000000538, 1000000539, 1000000540,
144     * 1000000541, 1000000542, 1000000543, 1000000544, 1000000545, 1000000546,
145     * 1000000547, 1000000548, 1000000549, 1000000550, 1000000551, 1000000552,
146     * 1000000553, 1000000554, 1000000555, 1000000556, 1000000557, 1000000558,
147     * 1000000559, 1000000560, 1000000561, 1000000562, 1000000563, 1000000564,
148     * 1000000565, 1000000566, 1000000567, 1000000568, 1000000569, 1000000570,
149     * 1000000571, 1000000572, 1000000573, 1000000574, 1000000575, 1000000576,
150     * 1000000577, 1000000578, 1000000579, 1000000580, 1000000581, 1000000582,
151     * 1000000583, 1000000584, 1000000585, 1000000586, 1000000587, 1000000588,
152     * 1000000589, 1000000590, 1000000591, 1000000592, 1000000593, 1000000594,
153     * 1000000595, 1000000596, 1000000597, 1000000598, 1000000599, 1000000600,
154     * 1000000601, 1000000602, 1000000603, 1000000604, 1000000605, 1000000606,
155     * 1000000607, 1000000608, 1000000609, 1000000610, 1000000611, 1000000612,
156     * 1000000613, 1000000614, 1000000615, 1000000616, 1000000617, 1000000618,
157     * 1000000619, 1000000620, 1000000621, 1000000622, 1000000623, 1000000624,
158     * 1000000625, 1000000626, 1000000627, 1000000628, 1000000629, 1000000630,
159     * 1000000631, 1000000632, 1000000633, 1000000634, 1000000635, 1000000636,
160     * 1000000637, 1000000638, 1000000639, 1000000640, 1000000641, 1000000642,
161     * 1000000643, 1000000644, 1000000645, 1000000646, 1000000647, 1000000648,
162     * 1000000649, 1000000650, 1000000651, 1000000652, 1000000653, 1000000654,
163     * 1000000655, 1000000656, 1000000657, 1000000658, 1000000659, 1000000660,
164     * 1000000661, 1000000662, 1000000663, 1000000664, 1000000665, 1000000666,
165     * 1000000667, 1000000668, 1000000669, 1000000670, 1000000671, 1000000672,
166     * 1000000673, 1000000674, 1000000675, 1000000676, 1000000677, 1000000678,
167     * 1000000679, 1000000680, 1000000681, 1000000682, 1000000683, 1000000684,
168     * 1000000685, 1000000686, 1000000687, 1000000688, 1000000689, 1000000690,
169     * 1000000691, 1000000692, 1000000693, 1000000694, 1000000695, 1000000696,
170     * 1000000697, 1000000698, 1000000699, 1000000700, 1000000701, 1000000702,
171     * 1000000703, 1000000704, 1000000705, 1000000706, 1000000707, 1000000708,
172     * 1000000709, 1000000710, 1000000711, 1000000712, 1000000713, 1000000714,
173     * 1000000715, 1000000716, 1000000717, 1000000718, 1000000719, 1000000720,
174     * 1000000721, 1000000722, 1000000723, 1000000724, 1000000725, 1000000726,
175     * 1000000727, 1000000728, 1000000729, 1000000730, 1000000731, 1000000732,
176     * 1000000733, 1000000734, 1000000735, 1000000736, 1000000737, 1000000738,
177     * 1000000739, 1000000740, 1000000741, 1000000742, 1000000743, 1000000744,
178     * 1000000745, 1000000746, 1000000747, 1000000748, 1000000749, 1000000750,
179     * 1000000751, 1000000752, 1000000753, 1000000754, 1000000755, 1000000756,
180     * 1000000757, 1000000758, 1000000759, 1000000760, 1000000761, 1000000762,
181     * 1000000763, 1000000764, 1000000765, 1000000766, 1000000767, 1000000768,
182     * 1000000769, 1000000770, 1000000771, 1000000772, 1000000773, 1000000774,
183     * 1000000775, 1000000776, 1000000777, 1000000778, 1000000779, 1000000780,
184     * 1000000781, 1000000782, 1000000783, 1000000784, 1000000785, 1000000786,
185     * 1000000787, 1000000788, 1000000789, 1000000790, 1000000791, 1000000792,
186     * 1000000793, 1000000794, 1000000795, 1000000796, 1000000797, 1000000798,
187     * 1000000799, 1000000800, 1000000801, 1000000802, 1000000803, 1000000804,
188     * 1000000805, 1000000806, 1000000807, 1000000808, 1000000809, 1000000810,
189     * 1000000811, 1000000812, 1000000813, 1000000814, 1000000815, 1000000816,
190     * 1000000817, 1000000818, 1000000819, 1000000820, 1000000821, 1000000822,
191     * 1000000823, 1000000824, 1000000825, 1000000826, 1000000827, 1000000828,
192     * 1000000829, 1000000830, 1000000831, 1000000832, 1000000833, 1000000834,
193     * 1000000835, 1000000836, 1000000837, 1000000838, 1000000839, 1000000840,
194     * 1000000841, 1000000842, 1000000843, 1000000844, 1000000845, 1000000846,
195     * 1000000847, 1000000848, 1000000849, 1000000850, 1000000851, 1000000852,
196     * 1000000853, 1000000854, 1000000855, 1000000856, 1000000857, 1000000858,
197     * 1000000859, 1000000860, 1000000861, 1000000862, 1000000863, 1000000864,
198     * 1000000865, 1000000866, 1000000867, 1000000868, 1000000869, 1000000870,
199     * 1000000871, 1000000872, 1000000873, 1000000874, 1000000875, 1000000876,
200     * 1000000877, 1000000878, 1000000879, 1000000880, 1000000881, 1000000882,
201     * 1000000883, 1000000884, 1000000885, 1000000886, 1000000887, 1000000888,
202     * 1000000889, 1000000890, 1000000891, 1000000892, 1000000893, 1000000894,
203     * 1000000895, 1000000896, 1000000897, 1000000898, 1000000899, 1000000900,
204     * 1000000901, 1000000902, 1000000903, 1000000904, 1000000905, 1000000906,
205     * 1000000907, 1000000908, 1000000909, 1000000910, 1000000911, 1000000912,
206     * 1000000913, 1000000914, 1000000915, 1000000916, 1000000917, 1000000918,
207     * 1000000
```


Projects

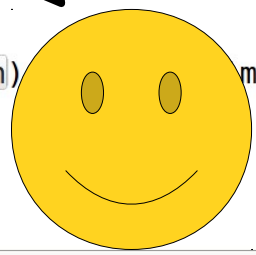
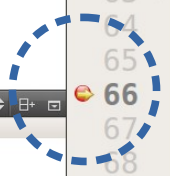
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51       * prime. These numbers were chosen because their product is less th
52       * 2027114911 = 1000000007 * 2027114911
53       */
54
55      int hashVal = 0;
56      for (char ch: first + last) {
57          /* Convert the input character to lower case. The numeric values
58           * of lower-case letters are always less than 127.
59           */
60          ch = tolower(ch);
61          hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
62      }
63      return hashVal;
64  }
65
66
67
68
69
70
71

```

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.



Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	65 'A'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

Projects

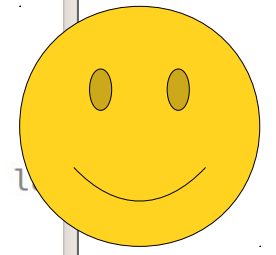
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- name_hash.cpp

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 l
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The num
64  * lower
65  */
66  ch =
67  hashV
68  }
69  return ha
70  }
71
```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st



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

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

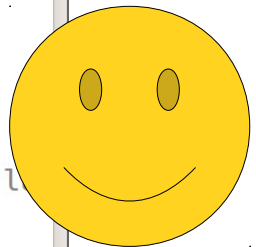
Open Documents

- name_hash.cpp

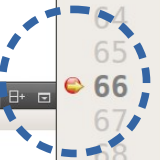
Qt

- Debug

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch = hashVal
67     }
68     return ha
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!



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1							
1	Main	name_hash...	31		nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Projects

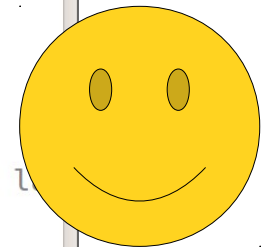
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- name_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch = hashVal
67     }
68     return ha
69 }
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st



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?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

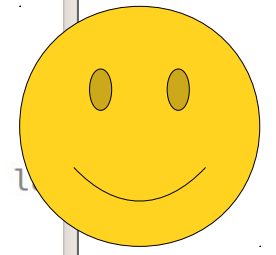
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- name_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch =
67         hashV
68     }
69     return ha
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st



The call stack can tell us exactly that!

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

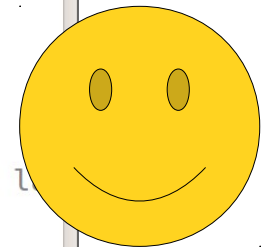
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- name_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch = hashV
67     }
68     return ha
69 }
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st



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 Main.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...

- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Projects

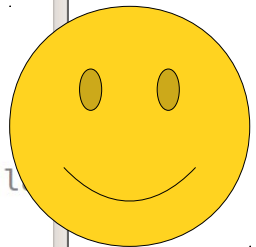
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 l
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The num
64  * lower
65  */
66  ch =
67  hashV
68  }
69  return ha
70  }
71

```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st



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

Open Documents

- name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

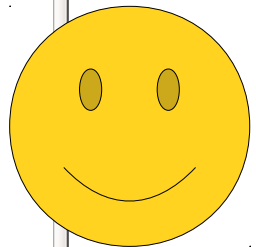
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name hash...	66								
1	Main	name hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	25								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...)

- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
18 #include "console.h"
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 implementation of the nameHash function. It
38 * to talk mo
39 * the meanti
40 * of the inp
41 *
42 * For those
43 * treats eac
44 * It then us
45 * F n where n is
```



• you'll end up over here!

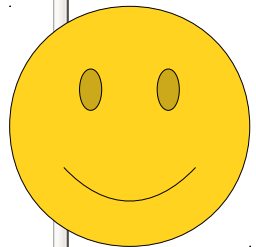
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
18 #include "console.h"
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 implementation of the nameHash function.
38 * to talk to the user and get the input.
39 * the meaning of the input.
40 * of the input.
41 *
42 * For those who are interested, the function
43 * treats each character as a number.
44 * It then uses a simple algorithm to
45 * find where the character is in the alphabet.
```



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

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

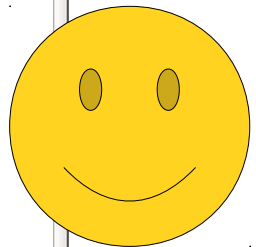
Open Documents

- name_hash.cpp

```

18 #include "console.h"
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 implementation of the nameHash function. It
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where n is

```



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!

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

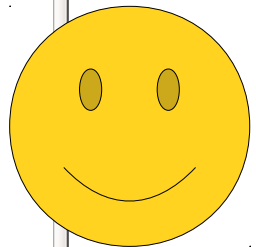
Open Documents

- name_hash.cpp

```

18 #include "console.h"
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 implementation of the nameHash function.
38  * to talk to the user and get their name.
39  * the meaning of the input string.
40  * of the input string.
41  *
42  * For those who are interested, the function
43  * treats each character of the input string
44  * as a number. It then uses a simple
45  * formula to calculate the hash value.

```



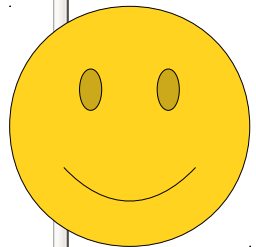
You might notice that there's some more stuff in the call stack beyond just main and nameHash. What are those?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, Open Documents, and Debug.

```
18 #include "console.h"
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 implementation of the nameHash function. It
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where n is
```



Let's find out! Double-click on the line marked "Main" on Level 2. When you do...

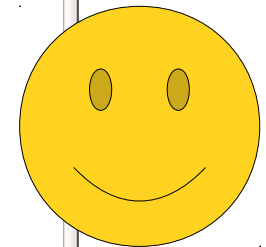
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



... you'll end up with something that looks like this.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

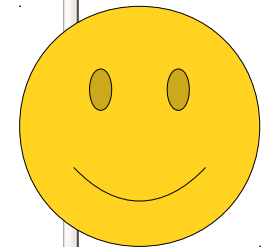
Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



Yikes! This looks hairy and scary! What happened?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

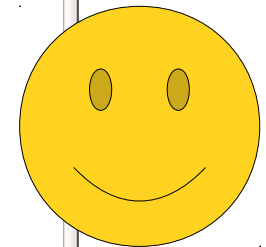
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



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.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

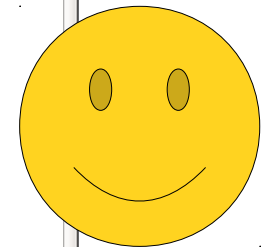
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



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

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

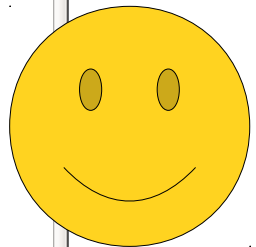
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



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.

Open Documents

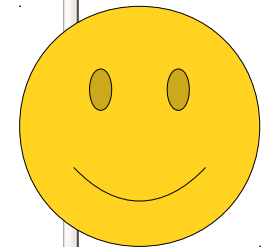
- main.cpp
- name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



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

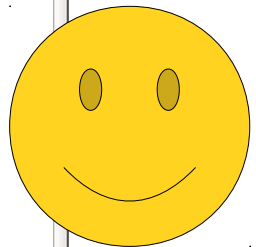
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



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

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

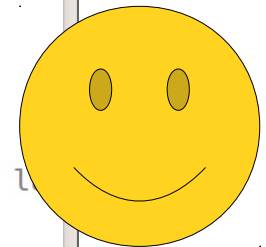
name-hash

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 l
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The num
64  * lower
65  */
66  ch =
67  hashV
68  }
69  return ha
70  }
71

```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st



You'll be teleported back to safety!

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal
58
59  /* It
60  * nam
61  */
62  for (c
63  /*
64  *
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

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



Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE interface showing a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeri
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	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing a C++ code editor, a call stack, and a variable watch window.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54
55
56
57
58
59
60
61
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeri
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 the following frames:

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

The variable watch window shows the following variables:

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Annotations:

- A speech bubble contains the text: "The call stack shows us how we got into the current function."
- A red arrow points from the speech bubble to the call stack entry for the current function (Level 0).
- A yellow smiley face is located to the right of the code editor.

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeri
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.



Open Documents

- main.cpp
- name_hash.cpp

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Look up at this panel over here.



Name	Value
__for_begin	@0x7fffffff030
__for_end	@0x7fffffff040
__for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

```
45
46
47 * but we
48 */
49
50 nameHash(string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49
50 nameHash(string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
▶ __for_begin	@0x7fffffff030
▶ __for_end	@0x7fffffff040
▶ __for_range	<not accessible>
ch	65 'A'
▶ first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
▶ last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

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.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
__for_begin	@0x7fffffff030
__for_end	@0x7fffffff040
__for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
__for_begin	@0x7fffffff030
__for_end	@0x7fffffff040
__for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	65 'A'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
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 las
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
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	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

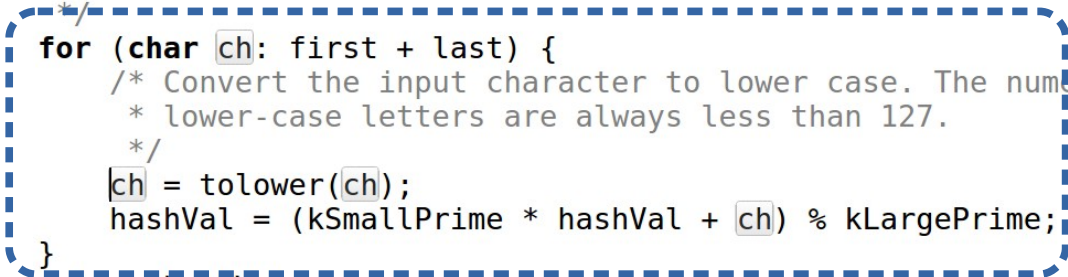
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
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 las
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
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	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	65 'A'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

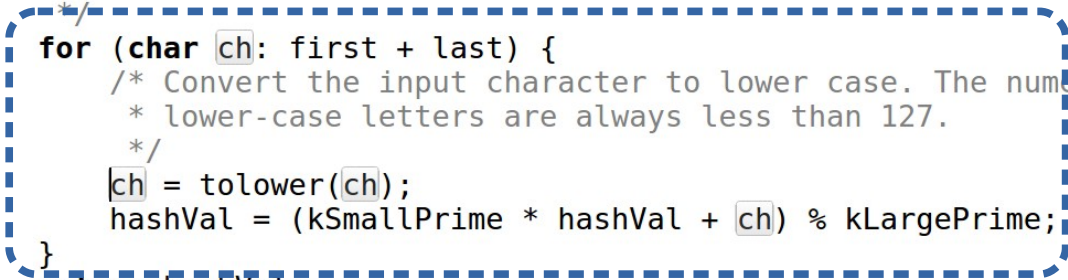
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49 nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
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 las
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
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	Type
__for_begin	@0x7fffffff030	std::string::const_iterator
__for_end	@0x7fffffff040	std::string::const_iterator
__for_range	<not accessible>	std::string::const_iterator
ch	65 'A'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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



```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
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 las
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
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
```

```
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
```

Name	Value	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	65 'A'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

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.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	not accessible	
ch	65 'A'	ch
first	@0x7ffffffe100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	65 'A'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name_hash.cpp', which implements a name hashing function. The code includes comments explaining the hashing scheme and defines two prime constants: `kLargePrime = 16908799` and `kSmallPrime = 127`. The function `nameHash` iterates over characters in the first and last names, converting them to lowercase and updating a hash value using the defined primes.

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 las
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
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 Qt stack trace window at the bottom shows the call stack for the `nameHash` function, with the current frame highlighted in red:

Level	Function	File	Line
0	nameHash	name_hash...	66
1	main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

A yellow smiley face icon is positioned to the right of the code editor. A speech bubble points to the stack trace window with the text: "Right above the stack trace, you'll see there are some small button icons." The Qt interface also shows a sidebar with various tool icons and a status bar at the bottom with tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.



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

Qt IDE interface showing a C++ code editor with a function definition for `nameHash`. The code includes comments and constants for prime numbers. A red box highlights a button in the bottom toolbar, and a speech bubble provides instructions on how to use it.

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 las
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
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 IDE interface showing a C++ code editor with a function definition for `nameHash`. The code includes comments and constants for prime numbers. A red box highlights a button in the bottom toolbar, and a speech bubble provides instructions on how to use it.

Qt IDE interface showing a C++ code editor with a function definition for `nameHash`. The code includes comments and constants for prime numbers. A red box highlights a button in the bottom toolbar, and a speech bubble provides instructions on how to use it.

Qt IDE interface showing a C++ code editor with a function definition for `nameHash`. The code includes comments and constants for prime numbers. A red box highlights a button in the bottom toolbar, and a speech bubble provides instructions on how to use it.



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."

Qt IDE interface showing a C++ project named 'name-hash'. The main editor window displays the source code for 'name_hash.cpp', which implements a simple polynomial hashing function. The code includes comments explaining the use of two prime numbers, kLargePrime (16908799) and kSmallPrime (127), and iterates over the characters of the input strings to calculate a hash value.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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
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 Qt IDE interface includes a sidebar with project files, a 'Debug' console at the bottom, and a 'Variables' window on the right showing memory addresses and values for variables like 'ch' and 'hashVal'. A yellow smiley face is drawn on the right side of the editor window.

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...

Qt IDE interface showing a C++ code editor with the following code:

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 las
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
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 IDE also shows a variable declaration table on the right:

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

A yellow smiley face is drawn on the right side of the code editor.

A speech bubble contains the text: "...your window should look something like this."

The bottom status bar shows: 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE interface showing a C++ code editor with a yellow smiley face and a callout bubble.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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
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 IDE interface showing a C++ code editor with a yellow smiley face and a callout bubble.

Open Documents

- main.cpp
- name_hash.cpp

Threads: #1 name-hash

Level	Function	File	Line
0	nameHash	name_hash...	67
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



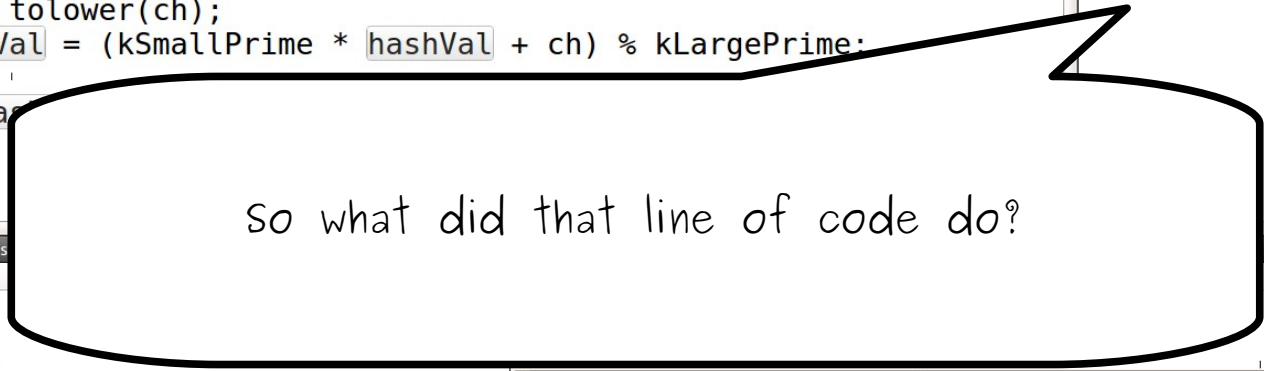
We're now at the line right after the one where we stopped. You just ran a single line of the program! Pretty cool!

Qt IDE interface showing a C++ code editor with a function `nameHash`. The code calculates a hash value for a string by iterating over its characters and applying a polynomial function. A yellow smiley face and a speech bubble are overlaid on the code.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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
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
```

so what did that line of code do?

Level	Function	File	Line
0	nameHash	name_hash...	67
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27



Qt IDE interface showing a C++ code editor with a nameHash function. The code includes comments about prime numbers and a loop that iterates over characters in two strings, updating a hash value. A specific line of code, `ch = tolower(ch);`, is highlighted with a blue box. A yellow smiley face is drawn next to the code, and a speech bubble points to the highlighted line with the following text:

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.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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
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
```

Level	Function	File	Line
0	nameHash	name_hash...	67
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

```

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

```

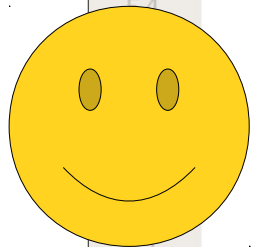
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;

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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

```

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



Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects: name-hash

- name-hash.pro
- Headers
- Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
- Other files

Open Documents:

- main.cpp
- name_hash.cpp

name-hash

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

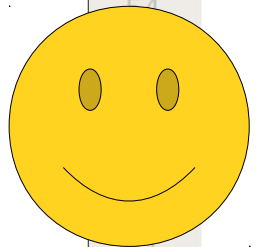
nameHash(string, string): int # Line: 67, Col: 9

```
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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

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



Name	Value	Type
for_begin	@0x7fffffff030	std::string
for_end	@0x7fffffff040	std::string
for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

```

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

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

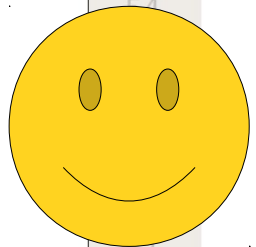
```

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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

```



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

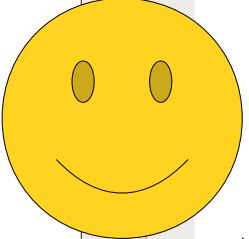
Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE sidebar containing icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, Open Documents, and Debug.

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.



```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

```
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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
}
```

Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

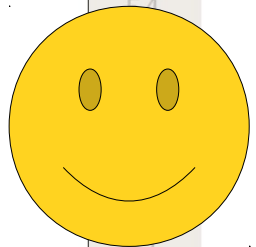
Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE sidebar containing various tool icons and panels: Welcome, Edit, Design, Debug, Projects, Analyze, Help, Open Documents, and Debug.

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



```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55 static const int kLargePrime = 16908799,
56 static const int kSmallPrime = 127;
57
58 int hashVal = 0;
59
60 /* Iterate across all the characters in the first name, then the las
61 * name, updating the hash at each step.
62 */
63 for (char ch: first + last) {
64     /* Convert the input character to lower case. The numeric values
65     * lower-case letters are always less than 127.
66     */
67     ch = tolower(ch);
68     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69 }
70 return hashVal;
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

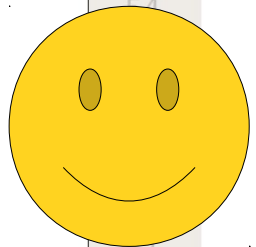
Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE sidebar containing icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, Open Documents, and Debug.

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.



```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

```
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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
}
```

Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

Debug

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  *
47  *
48  *
49  *
50  *
51  *
52  *
53  *
54  *
55  *
56  *
57  *
58  *
59  *
60  *
61  *
62  *
63  *
64  *
65  *
66  *
67  *
68  *
69  *
70  *
71  *

```

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

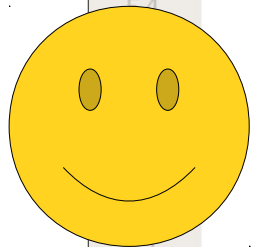
```

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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

```



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

Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

Debug

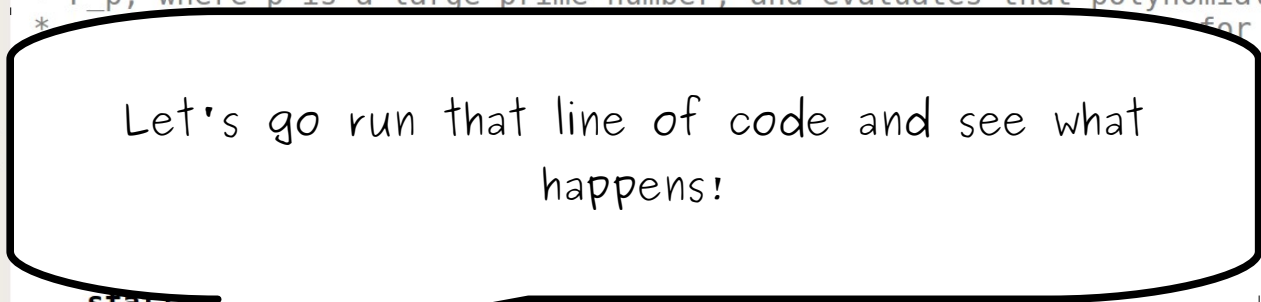
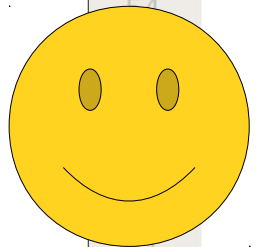
```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

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

```
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
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```



hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

Name	Value	Type
__for_begin	@0x7fffffff030	std::string::basic_string_iterator
__for_end	@0x7fffffff040	std::string::basic_string_iterator
__for_range	<not accessible>	std::string::basic_string_iterator
ch	97 'a'	char
first	@0x7fffffff100	std::string::basic_string_iterator
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string::basic_string_iterator

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE interface showing a C++ code editor with the following code:

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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
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 code editor shows a red box around the "Step Over" button in the debug toolbar. A yellow smiley face is positioned to the right of the code editor. A speech bubble points to the "Step Over" button with the text:

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

The right sidebar shows a memory dump table:

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	st
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Qt IDE interface showing a C++ code editor with a function definition for `nameHash`. The code calculates a hash value based on two prime numbers, `kLargePrime` and `kSmallPrime`.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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 val
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 }
```

A yellow smiley face is drawn next to the code. A speech bubble points to the code with the text: "... you'll end up with something like this!".

The Qt IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main window shows the code editor, a variable declaration table, and a stack trace.

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	st
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Level	Function	File	Line
0	nameHash	name_hash...	62
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt Creator IDE interface showing a C++ project named 'name-hash'. The main editor window displays the source code for 'name_hash.cpp', with the current cursor position at line 62, column 5. The code defines a function 'nameHash' that calculates a hash value for a string. The function iterates over each character, converting it to lowercase and updating the hash value using a large prime number 'kLargePrime' and a small prime number 'kSmallPrime'. The hash value is calculated as $hashVal = (kSmallPrime * hashVal + ch) \% kLargePrime$.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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 val
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 'Open Documents' panel shows 'main.cpp' and 'name_hash.cpp'. The 'Debug' panel at the bottom shows the execution stack with the following entries:

Level	Function	File	Line
0	nameHash	name_hash...	62
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

The 'Name' panel on the right shows memory addresses and values for various variables:

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	st
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

A yellow smiley face is drawn on the right side of the screen, with a speech bubble pointing to the code. The speech bubble contains the text: "Let's see what's changed."

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!



```
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  
    * lower-case letters are always less than 127.  
    */
```

```
    ch = tolower(ch);  
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
```

```
}  
return hashVal;
```

```
}
```

Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

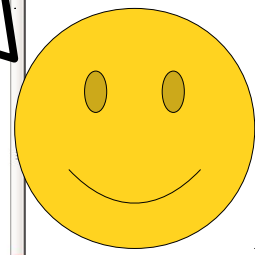
Open Documents

- main.cpp
- name_hash.cpp

name-hash

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - 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 range [first, last) and compute
60     * the hash value.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric value
64         * of 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
```

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	Type
__for_begin	@0x7fffffff030	std::string::basic_string_iterator
__for_end	@0x7fffffff040	std::string::basic_string_iterator
__for_range	<not accessible>	std::string::basic_string_iterator
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

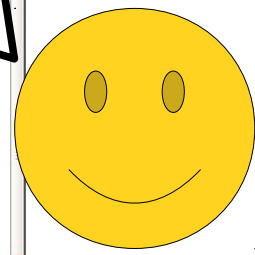
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt
Welcome
Edit
Design
Debug
Projects
Analyze
Help

Open Documents
main.cpp
name_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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 range [first, last)
60     * name,
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric value
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 }
```

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



Name	Value	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

Qt

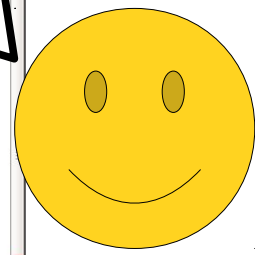
Debug

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 range [first, last)
60  * name,
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value 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

```

Let's go do it again!



Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

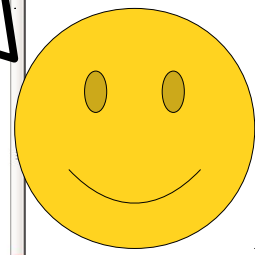
Open Documents

- main.cpp
- name_hash.cpp

name-hash

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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     * name,
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric value
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 }
```

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
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

name-hash

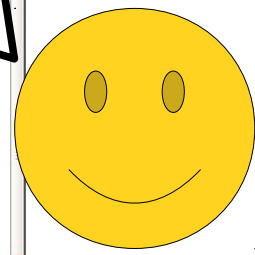
```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 name,
60  * name,
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value 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	Type
for_begin	@0x7fffffff030	std::string
for_end	@0x7fffffff040	std::string
ch	<not accessible>	char
hashVal	100 'd'	int
kLargePrime	@0x7fffffff100	int
kSmallPrime	16908799	int
last	127	int
first	@0x7fffffff120	std::string

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



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

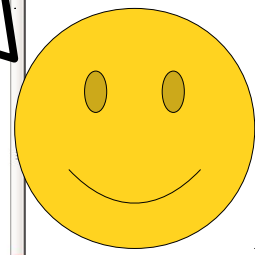
- name-hash
- name-hash.pro
- Headers
- Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
- Other files

Open Documents

- main.cpp
- name_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
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
51     * prime. These numbers were chosen because their product is less th
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     * name,
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric value
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
```

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



Name	Value	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	100 'd'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

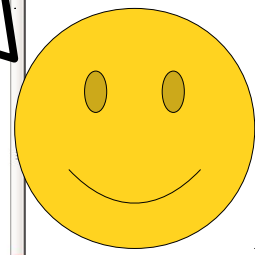
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16999799;
55  static const int kSmallPrime = 97;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the string.
60  * name, last, first.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value 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

```

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'.



Open Documents

- main.cpp
- name_hash.cpp

Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	100 'd'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16998799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

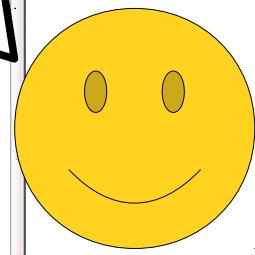
- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16999799
55  static const int kSmallPrime = 97
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the range [first, last)
60  * name, first, last)
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value 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, click "Step Over" one more time.



Open Documents

- main.cpp
- name_hash.cpp



Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

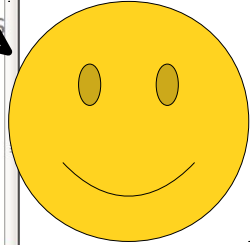
Qt IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name_hash.cpp". A large speech bubble is overlaid on the code, containing a handwritten instruction. The code includes a function "nameHash" that calculates a hash value based on the first and last characters of a string. The variable "hashVal" is updated in a loop. A debugger window at the bottom shows the execution stack.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int
50
51
52
53
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63      /* Convert the input character to lower case. The numeric
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
```

Debugger Stack:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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!



Name	Value	Type
__for_begin	@0x7fffffff030	stc
__for_end	@0x7fffffff040	stc
__for_range	<not accessible>	
ch	100 'd'	ch
first	fffffe100	stc
hashVal	?	int
kLargePrime	99	int
kSmallPrime	127	int
last	@0x7fffffff120	stc

Qt Creator IDE interface showing a C++ code editor, a variable watch window, and a debugger console.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric
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
```

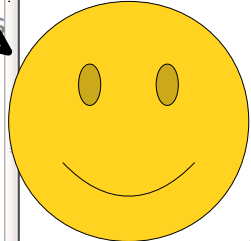
Variable Watch Window:

Name	Value	Type
__for_begin	@0x7fffffff030	stc
__for_end	@0x7fffffff040	stc
__for_range	<not accessible>	
ch	100 'd'	ch
first	ffffffe100	stc
hashVal	?	int
kLargePrime	99	int
kSmallPrime	127	int
last	@0x7fffffff120	stc

Debugger Console:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

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.



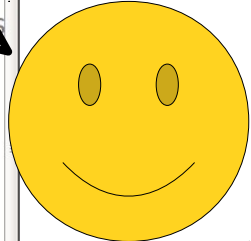
Qt IDE interface showing a C++ code editor with a breakpoint at line 62. A speech bubble explains that the breakpoint should clear. A yellow smiley face is also present.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric
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 IDE interface showing a C++ code editor with a breakpoint at line 62. A speech bubble explains that the breakpoint should clear. A yellow smiley face is also present.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

• 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!



Qt IDE interface showing a C++ code editor with a function `nameHash`. The code includes comments and constants for prime numbers. A red box highlights the `return hashVal;` statement on line 69. A yellow smiley face is drawn next to the code. A speech bubble contains the text: "Now, take a look back at these buttons." The bottom status bar shows "1 Issues 2 Search Results 3 Application O".

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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 v
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  }
```

Level	Function	File	Line
0	nameHash	name_hash...	62
1	nameHash	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt IDE screenshot showing a C++ code editor with a function `nameHash`. The code iterates over characters in two strings and calculates a hash value. A red box highlights the `return hashVal;` statement on line 69. A yellow smiley face is drawn next to the code, with a speech bubble pointing to it.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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 v
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  }
```

Threads: #1 name-has

Level	File	Line
0	nameHash	name_hash... 62
1	Main	name_hash... 31
2	Main	main.cpp 23
3	startupMain	platform.cpp 2208
4	main	name_hash... 27

1 Issues 2 Search Results 3 Application O

Hover your mouse over the one that's fifth from the left. When you hover over it, it should say "step out."

Qt IDE interface showing a C++ code editor with the following code:

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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 v
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 Qt IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The 'Open Documents' list shows 'main.cpp' and 'name_hash.cpp'. The 'Threads' window at the bottom shows a stack trace with a red box around the 'nameHash' function call at line 62. The 'Variables' window on the right shows a table with a red box around the 'hashVal' variable, which has a question mark icon next to its value.

Name	Value	Type
__for_begin	@0x7fffffff030	std::string::size_type
__for_end	@0x7fffffff040	std::string::size_type
__for_range	<not accessible>	std::string::size_type
ch	100 'd'	char
first	fffffe100	std::string
hashVal	?	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

A yellow smiley face is drawn next to the code. A speech bubble points to the 'nameHash' function call in the stack trace, containing the text: "If you click this button, it will keep running this function up until it completes and returns."

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name_hash.cpp', which implements a simple polynomial hash function. The code is as follows:

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
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
51  * prime. These numbers were chosen because their product is less th
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 las
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 v
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 Qt IDE interface includes a sidebar with project files, a 'Debug' console at the bottom, and a 'Variables' window on the right. The 'Variables' window shows the current state of variables, with 'hashVal' highlighted and its value set to 99. A red box highlights the 'hashVal' variable in the 'Variables' window, and a yellow smiley face is drawn next to it. A speech bubble points to the 'hashVal' variable in the code, containing the text: 'Now, go click that button. If you did everything right...'

Name	Value	Type
__for_begin	@0x7fffffff030	st
__for_end	@0x7fffffff040	st
__for_range	<not accessible>	
ch	100 'd'	ch
first	fffffe100	st
hashVal	99	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	st

Level | File | Line

0	nameHash	name_hash... 62
1	Main	name_hash... 31
2	Main	main.cpp 23
3	startupMain	platform.cpp 2208
4	main	name_hash... 27

1 Issues 2 Search Results 3 Application O

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

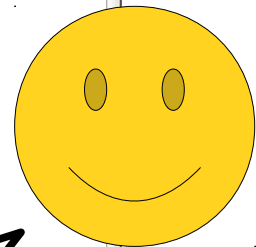
Open Documents

- main.cpp
- name_hash.cpp

```

18 #include "console.h"
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 act
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where p

```



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

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

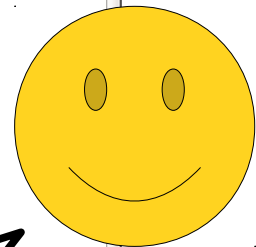
- main.cpp
- name_hash.cpp

```

18 #include "console.h"
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 act
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where p

```

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613



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

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

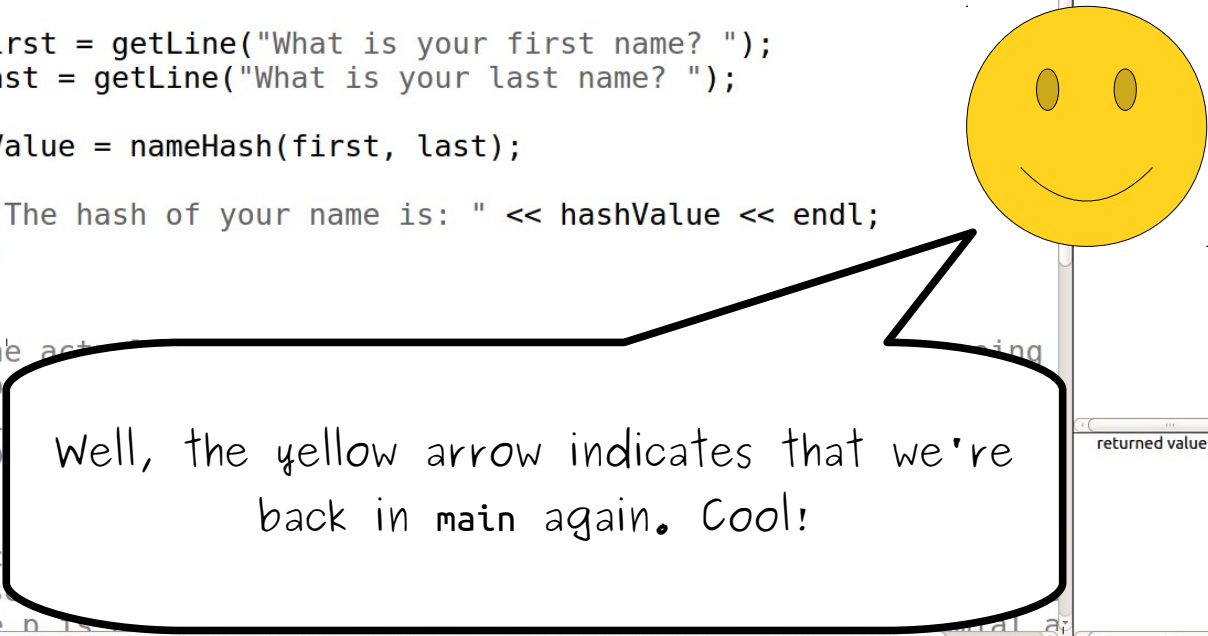
Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
18 #include "console.h"
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 implementation of the nameHash function. It
38  * to talk more about the meaning of the input parameters.
39  * For those who are interested, the function treats each character
40  * It then uses a simple algorithm to calculate the hash value.
41  * For more information, see the documentation for the nameHash function.
```

Qt IDE right sidebar showing a variable viewer with a table of variables and their values.

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457



Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
 - name-hash.pro
 - Headers
 - Sources
 - lib/StanfordCPPLib
 - src
 - name_hash.cpp
 - Other files

Open Documents

- main.cpp
- name_hash.cpp

```
18 #include "console.h"
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     last = getLine("What is your last name? ");
30
31     hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34 }
35
36
37 /* This is the actual implementation of the nameHash function. It
38 * to talk to the user and get their first and last names.
39 * the meaning of the input strings.
40 * of the input strings.
41 *
42 * For those who are interested, the nameHash function
43 * treats each character as a number from 0 to 255.
44 * It then uses a simple algorithm to calculate the hash value.
45 * For example, if the first name is "John" and the last name is "Doe",
46 * the hash value would be 1967457.
```



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

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								



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.)

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name_hash.cpp'. The code includes a namespace 'std', a function prototype for 'nameHash', a 'main' function that prints the hash value, and a detailed comment explaining the hash function's algorithm. The 'Debug' console at the bottom shows the execution flow, and the 'Variables' window on the right shows the current state of variables.

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for
23  * in main and
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
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
```

Variables window:

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

Threads window:

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

Debug console: returned value 1967457



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

```
int hashValue = nameHash(first, last);
```

```
18 #include "console.h"
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for
23  * in main and
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
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 character
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
```

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								



This is pretty cool, actually!

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name_hash.cpp'. A blue box highlights the line: `int hashValue = nameHash(first, last);`. The right sidebar shows a variable inspector with 'first' and 'last' variables. The bottom status bar shows the thread stack.

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for
23  * in main and
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. 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
```

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name_hash.cpp'. A blue box highlights the line: `int hashValue = nameHash(first, last);`. The right sidebar shows a variable inspector with 'first' and 'last' variables. The bottom status bar shows the thread stack.

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27



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!

```
int hashValue = nameHash(first, last);
```

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for
23 * in main and
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
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 character
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
```

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								



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 IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name_hash.cpp".

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and
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 hashCode = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashCode << 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
```

The Qt console shows the output: "The hash of your name is: 1967457".

The Debug Console shows the following stack trace:

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

The status bar at the bottom shows: "Threads: #1 name-hash Stopped: "function-finished"."



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

```
File Edit Build Debug Analyze Tools Window Help
Projects
name-hash
  name-hash.pro
  Headers
  Sources
    lib/StanfordCPPLib
    src
  name_hash.cpp
  Other files
name-hash.cpp
main.cpp
name_hash.cpp

20 using namespace std;
21
22 /* Prototype for the nameHash function. This is used
23  * in main and the nameHash function.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
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
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 CS
47  * but we thought it might be fun!

Name Value Type
first @0x7fffffff0a0 std::string
last @0x7fffffff0c0 std::string
hashValue 1967457 int

Threads: #1 name-hash Stopped: "end-stepping-range".
Level Function File Line Number Function File Line Address Condition Ignore Threads
0 Main name_hash... 33
1 Main main.cpp 23
2 startupMain platform.cpp 2208
3 main name_hash... 27

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages
```




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.

```
File Edit Build Debug Analyze Tools Window Help
Projects
name-hash
  name-hash.pro
  Headers
  Sources
    lib/StanfordCPPLib
    src
  name_hash.cpp
  Other files
name_hash.cpp
20 using namespace std;
21
22 /* Prototype for the nameHash function, which is used
23  * in main and the nameHash function.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
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 CS
47  * but we thought it might be fun!
```

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	1967457	int

Open Documents

- main.cpp
- name_hash.cpp

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	33								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt Creator IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name_hash.cpp".

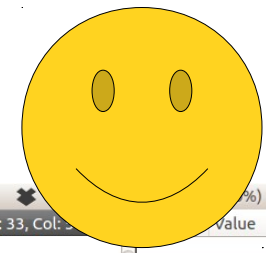
```
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string 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. We're going
38  * to talk more about what hash functions do later. In the meantime,
39  * think of it as a function that takes the character of the input
40  * and produces a number.
41  */
```

The Qt icon in the bottom-left toolbar is highlighted with a yellow box. A speech bubble points to it with the text: "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.'" A yellow smiley face is drawn next to the speech bubble.

On the right side, a variable viewer shows:

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	1967457	int

At the bottom, the status bar shows tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.



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!

```
20 int main() {
21     st
22     st
23     in
24     co
25     re
26 }
27
28 /* This
29 * to
30 * the
31 * of
32 * For
33 * tre
34 * It
35 * F_p, where p is a large prime number, and evaluates that polynomial a
36 * some smaller prime number q. (You aren't expected to know this for CS
37 * but we thought it might be fun!
```

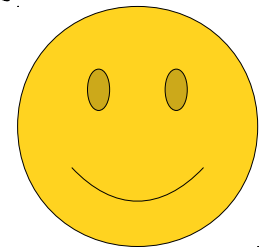
```
File Edit Options Help
What is your first name? Ada
What is your last name? Lovelace
The hash of your name is: 1967457
```

Threads: Debugger finished.

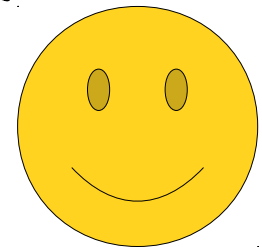
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

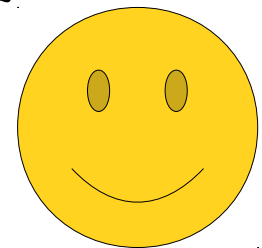
so there you have it! You've now gotten more familiar with the debugger!



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



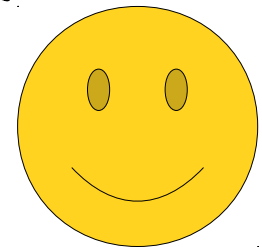
You know how to read the call stack and to see the values of local variables.



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



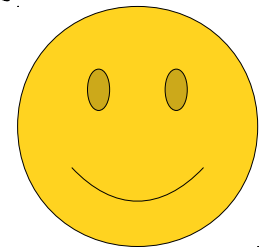
You know how to run a function to completion,
and how to let the program keep on running.



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!

