Week 5 Tuesday
Generics I

void*. Why'd it have to be void*?
Starter Code

git clone /afs/ir/class/archive/cs/cs107a/cs107a.1226/WWW/exercises/generics1
Announcements

- Midterm is next Tuesday
  - Date/Time: Tuesday, May 3rd, 7pm-9pm
  - Locations: 420-040 and STLC 111
  - See more info on the CS 107 website
  - If you have a conflict, email Nick by Wed

- assign4 schedule

- CS 107A Midterm Review Session next Sunday, time/location TBD
  - Practice Exam released later
Unix Tip Spotlight

- Exiting gdb, ssh sessions, terminals
  - `<CTRL+D>`: send EOF to the current program
  - Causes gdb, ssh, shells, etc to quit
Agenda

- void*
- struct pair
void*
What is a `void*`?

- A `int*` is a pointer to 1+ values of type ____
- A `char**` is a pointer to 1+ values of type ____
- A `struct foo*` is a pointer to 1+ values of type ____
- A `void**` is a pointer to 1+ values of type ____
- A `void*` is a pointer to 1+ values of type ____
What is a `void*`?

- A `int*` is a pointer to 1+ values of type `int`
- A `char**` is a pointer to 1+ values of type ____
- A `struct foo*` is a pointer to 1+ values of type ____
- A `void**` is a pointer to 1+ values of type ____
- A `void*` is a pointer to 1+ values of type ____
What is a `void*`?

- A `int*` is a pointer to 1+ values of type `int`
- A `char**` is a pointer to 1+ values of type `char*`
- A `struct foo*` is a pointer to 1+ values of type `___`
- A `void**` is a pointer to 1+ values of type `___`
- A `void*` is a pointer to 1+ values of type `___`
What is a `void`*?

- A `int`* is a pointer to 1+ values of type `int`
- A `char`** is a pointer to 1+ values of type `char`*
- A `struct foo`* is a pointer to 1+ values of type `struct foo`
- A `void`** is a pointer to 1+ values of type ____
- A `void`* is a pointer to 1+ values of type ____
What is a `void*`?

- A `int*` is a pointer to 1+ values of type `int`
- A `char**` is a pointer to 1+ values of type `char*`
- A `struct foo*` is a pointer to 1+ values of type `struct foo`
- A `void**` is a pointer to 1+ values of type `void*`
- A `void*` is a pointer to 1+ values of type ____
What is a `void*`?

- A `int*` is a pointer to 1+ values of type `int`
- A `char**` is a pointer to 1+ values of type `char*`
- A `struct foo*` is a pointer to 1+ values of type `struct foo`
- A `void**` is a pointer to 1+ values of type `void*`
- A `void*` is a pointer to 1+ values of type `void???
What is a void*?

- A int* is a pointer to 1+ values of type int
- A char** is a pointer to 1+ values of type char*
- A struct foo* is a pointer to 1+ values of type struct foo
- A void** is a pointer to 1+ values of type void*
- A void* is a pointer to 1+ values of type void??
  - But there is no such thing as a value of type void!
What is a `void*`?

- Values of type `void*` are pointers to something, but it's not specified what it is ("it's a `void`, but that doesn't exist!")
- We use them when we don't want to commit to specifying what the pointer points to while writing code e.g. for generics
  - But when the function is actually run, something real like an `int` is being pointed to

```c
void qsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void *));
```
What can’t you do with `void*`?

- You can’t do anything that gets what it points to
  - You cannot dereference it e.g. `*ptr`
  - You cannot index into it e.g. `ptr[0]`
- You can’t do pointer arithmetic, since that requires the size of what it points to e.g. `ptr + 3`
  - But you can cast to something else first, like `(char*)ptr + 3`
- The below is illegal and does not compile!

```c
void get_3rd_elem(void *arr, size_t n) {
    return arr[2];
}
```
void swap(void *a, void *b, size_t elem_size);

= = = = = = Generic Function Boundary = = = = = =

Code dealing with strings

Code dealing with integers
Generic Functions

void swap(void *a, void *b, size_t elem_size);

Code here can only assume it’s working on pointers to something!

= = = = = = Generic Function Boundary = = = = = =

Code dealing with strings

Code here can assume it’s working with strings!

Code dealing with integers

Code here can assume it’s working with ints!
What can you actually do with `void*`?

- If given nothing:
  - You can’t really do anything
- If given `elem_size`:
  - You can copy elements
    - `void *memcpy(void *dest, const void *src, size_t n);`
    - `void *memmove(void *dest, const void *src, size_t n);`
  - You can get a pointer to a later element, where `i < num_elems`
    - `(char*)ptr + i*elem_size`
- If given a function pointer that operates on the `void*`, can use that
struct pair
struct pair

on the stack

Stack

two_elem_arr   elem_size
------         -----
    7           4

Heap

two_elem_arr   elem_size
------         -----
  hello"      "world"
struct pair on the heap

Stack

Heap

struct pair {
    void *two_elem_arr;
    size_t elem_size;
};

int_pair_ptr

str_pair_ptr

two_elem_arr
elem_size 4

7 6

two_elem_arr
elem_size 8

"hello"
"world"
struct pair {
    void *two_elem_arr;
    size_t elem_size;
};

Stack:
- p
- first
- second
- elem_size

Memory being pointed to, idk where it is

Heap

pair_create initial

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>two_elem_arr</td>
<td>?</td>
</tr>
<tr>
<td>elem_size</td>
<td>?</td>
</tr>
</tbody>
</table>

7
6
4
```c
struct pair {
    void *two_elem_arr;
    size_t elem_size;
};
```

Memory being pointed to, idk where it is

**Stack**
- `p`
- `first`
- `second`
- `elem_size`

**Heap**
- `two_elem_arr`
- `elem_size: 4`
- `7 6`
- `7`
- `6`