← Lab 5: Advanced pointers

Due by midnight, Wednesday 10/24

> If this lab is confusing right now, we'll be talking more about this stuff on Monday.

Pointers are super important when writing low-level programs! GET COMFORTABLE WITH EM!

In this lab, you'll be writing a **generic array-filtering function.** This will make use of pointer arithmetic, function pointers, and pointer casting. This is an actually useful function! Filtering values out of an array is a very common operation.

Starting off

Get the starting file here. Rename it properly, and upload to thoth.

> Open it and read the comments. All of them.

Predicates

"Predicate" is a common programming term which means "something that gives a yes-or-no answer."

The `filter` function's predicate must be a function which:

- takes a `const void*` which points to **a value from the array**
- returns an integer:
  - 0 for **false** (ignore the item)
  - nonzero for **true** (put the item in the output array)
    - (This is common in C, because it didn't use to have `bool`.)

Writing the predicate function

The `less_than_50` function should *interpret its parameter as a pointer to a* `float`, and as the name implies, return a "true" (nonzero) value if it is less

than 50.

Since the parameter is a `const void*`, you'll have to cast the parameter to a different pointer type.

Have a look at **how I wrote the comparison function in the `qsort.c` example** to get an idea of how to write this.

> Hint: in C, comparison operators give an integer value. They give 1 if they're true, and 0 if they're false.

Writing the `filter` function

You didn't read the comments, did you. 🧠

Have a look at the code in `main`:

```
float filtered[NUM_VALUES];
int filtered_len = filter(filtered, float_values, NUM_VALUES,

printf("there are %d numbers less than 50:\n", filtered_len);

for(int i = 0; i < filtered_len; i++)
    printf("\t%.2f\n", filtered[i]);
```

Look at the `float_values` array and think about what the output *should* look like. (There are 6 numbers less than 50, right?)

The `filter` function should work like this:

- for each item in the `input` array:
  - call the `pred` function with a pointer to that element
  - if it returned "true":
    - use `memcpy` to copy that item from the input array to the output array (see below)

In addition it should:

- keep a count of how many items "passed the test" (predicate returned "true")

- return that count

`memcpy`

`memset` is used to fill in a blob of bytes with a value. `memcpy` is used to copy blobs of bytes from one place to another. It's a very common function.

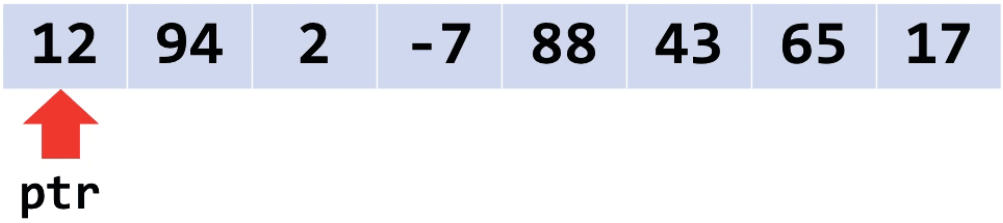
```
memcpy(dest, src, length);
```

This will copy `length` bytes from the memory pointed to by `src` into the memory pointed to by `dest`.

"Walking pointers"

You're used to using `[]` to access values from arrays. But you can't use `[]` on a `void*`. Instead, an easier technique is to use a "walking pointer."

Instead of keeping a pointer to the beginning of an arrays, we **move the pointer along, item by item, to access the array.** Like this.



But there's a catch: **you can't do pointer arithmetic on void pointers either!!**

So if you want to move a `void*` over by *n* bytes, you have to:

- cast it to a `char*`
- add *n* to that
- store it back into the `void*`

All this can be done on **one line.** Don't overcomplicate things.

Good luck, but some likely mistakes:

If you **don't move the pointer along the `input` array,** you'll get something like:

```
there are 10 numbers less than 50:
        31.94
        31.94
        31.94
        ...etc...
```

If you **don't move the pointers by the right number of bytes,** you might get something like:

```
there are 6 numbers less than 50:
        127.76
        36.10
        0.00
        ...etc...
```

If you moved the `input` pointer right, but **forgot to move the `output` pointer along:**

```
there are 6 numbers less than 50:
        19.60
        0.00
        ...etc...
```

If you didn't **count properly,** or maybe you didn't **respond to the predicate properly:**

```
there are 0 numbers less than 50:
```

Submission

Please make sure the driver (the `main` function) is the default one I gave you before you submit.

Then submit as usual.