Pointers

- * is the dereference operator
- & is the reference operator
- Example: int swap function
  - void charSwap(char * a, char * b){
    - char temp;
    - temp = *a;
    - *a = *b;
    - *b = temp;}
  - char c1 = ‘a’;
  - char c2 = ‘b’;
  - charSwap(&c1, &c2);
An array is a collection of items in memory together, which can be used to store related data. To declare an array of 10 integers, you would write:

```c
int arr[10];
```

Strings are arrays of characters and are NULL terminated. Instead of an array of characters, you can declare a string:

```c
char str[10];
```

2D arrays are also supported in C. To declare a 2D array of 10 rows and 10 columns, you would write:

```c
int 2d_arr[10][10];
```

The formula for accessing elements in a 2D array is `row * num_cols + cols`.
“Recursion” and “Backtracking”

- Recursion generally involves using a solution to a more basic problem, and a reductive step with base case implementation to ultimately deliver the answer to the caller.
  - In general, it can function somewhat as a loop

- Backtracking is a generally recursive use of a global/shared resource to produce / find a particular result
  - In general involving permuting/modifying that static resource in some way
  - Then checking if that permutation actually worked, if yes, return the result
  - Else: backtrack, and undo that permutation
Types

- Cool things about types in C
  - Types can be changed dynamically with a cast! ie (5/7)==0, but (((float) 5)/7)!=0
  - Unsigned vs Signed! Unsigned vars will Zero-EXT, Signed vars Sign-EXT, when casted to a longer data type ((int) ((char) -5)) != ((unsigned int) ((unsigned char) -5)), when doing comparison operations (<, >, etc.)
  - Why does the type of a pointer matter? The amount of bytes that are returned when a pointer is dereferenced, The amount the actual pointer value changes when a quantity is added to it.
  - Array pointer duality: Where arrays are represented by pointers to the first element of an array, and can be dereferenced and incremented in order to access other elements.
  - char alf[4] = “abc” is the same as char alf[4] = {‘a’, ‘b’, ‘c’, 0} (the Null character is automatically there in the first case because the compiler knows you need one) HOWEVER! If declared char * alf = “abc”; alf could point to read only memory, so it is non-writable.
Structs

- Cool things about structs in C
  - They basically just store memory offset information for the C compiler to use to look up particular data
  - `typedef struct point_t {int x; int y;} point_t;` //allows you to use point as a datatype
  - Using typedef adds point to the list of types, otherwise you would need to write struct point_t instead of point_t when using that data type
Run-Time Stack

Important Registers:
- R4: Global data section
- R5: Base of runtime stack
- R6: Top a runtime stack
- R7: Return address

Updating the runtime stack

Push:
ADD R6, R6, #-1 ; Update pointer
STR R0, R6, #0 ; Data in R0

Pop:
ADD R6, R6, #1 ; Update pointer
Run-Time Stack (Continued)

**Bookkeeping Info:**
- Callee frame pointer
- Return address
- Return value

**Things to Remember:**
- Activation record: how to create and tear-down
- The record is popped when exiting the function
- Update the stack frame
- Parameters pushed right to left
- Local variables pushed to stack

Don't let your memes be dreams
Cheat Sheet

- C to LC-3 example (check Patt and Patel)
- Examples of your weakest areas
  - Pointer Examples
  - Linked List Examples
  - Tree Traversals
  - Backtracking
  - Syntax?
- You have a lot of space
Practice Problems

https://goo.gl/RauPo4