Quick notes

- Check Piazza
- Reminders:
  - Assign4 is due friday
  - Python and diff/patch
  - Assign3 was really hard!
- If you guys are stressed out:
  - CAPS (https://www.counseling.ucla.edu/)
    - Free with UC ship
- Questions?

Feedback / Office Hours

- Tameez Latib
  - tameezlatib@gmail.com, please add “CS35L” to the subject line
  - Office Hours: Monday **4pm-6pm** (or by appointment)
  - Feedback: https://forms.gle/6kcJ2aJtzAzFMhHQ7 (anonymous google form)

C

- Like C++ but without classes
  - Have structs
- No strings, bools
  - Cstrings, 0/1
- Pointers + memory allocation
- Printf (no cin / cout)
- To compile: gcc file.c
  - On windows: need to install mingw64
  - set environment variable PATH to location of bin folder

Using printf (fast debugging)

- printf(“num: %d, char: %c, string: %s”, num, char, str)
- Warning:
  - printf(“Char: %s”, char) will cause a segmentation fault
  - You are trying to view ‘char’ as a pointer, so you go to the address of ‘char’, which is invalid

```c
#include <stdio.h>


int main(int argc, char *argv[]) {
    int n = 5;
    char c = 'H';
    char cstr[] = "This is a C string";
    printf("Number: %d. Character: %s, String: %s", n, c, cstr);
}
```

Almost-classes (structs)

- Can’t have ‘class’ methods
- Not object oriented
- Struct ~ group of variables

```c
#include <stdio.h>

struct Location
{
    float latitude, longitude;

};

void print_loc(struct Location place){
    printf("Located at latitude %f and longitude %f", place.latitude, place.longitude);

};

int main(int argc, char *argv[]) {
    struct Location UCLA;
    UCLA.latitude = 34.068921;
    UCLA.longitude = -118.4451811;
    print_loc(UCLA);
}
```

Pointers

- * ~ value, & ~ address
- Exercise:
  - What do these output?

```c
int main(int argc, char *argv[]) {
    int val = 10;
    int * valptr = &val;
    *valptr = 5;
    printf("valptr: %d. val: %d.", *valptr, val);
}
```

```c
int main(int argc, char *argv[]) {
    int * valptr;
    *valptr = 5;
    printf("valptr: %d", *valptr);
}
```

Pointers, malloc

- Need to allocate memory
- And then free memory
- Can also use calloc
  - Sets allocated memory to 0
  - Slower
- Realloc
  - Re-allocate memory
  - In case you need more

```c
int main(int argc, char *argv[]) {
    int * valptr;
    *valptr = 5;
    printf("valptr: %d", *valptr);
}
```

```c
int main(int argc, char *argv[]) {
    int * valptr;
    valptr = malloc(sizeof(int));
    *valptr = 5;
    printf("valptr: %d", *valptr);
    free(valptr);
}
```

Pointers, calloc / realloc example

- Note a[i] and *(a+i) the same
- Realloc does not change values in num_array
- output:

```
i: 0. arr[i]: 0
i: 1. arr[i]: 500
i: 2. arr[i]: 1000
i: 0. arr[i]: 0
i: 1. arr[i]: 500
i: 2. arr[i]: 1000
i: 3. arr[i]: 1500
```

```c
int main(int argc, char *argv[]) {
    int n = 3;
    int * num_array;
    num_array = calloc(n, sizeof(int));
    num_array[1] = 500;
    *(num_array+2) = 1000;
    for (int i = 0; i < n; i++){
        printf("i: %d. arr[i]: %d\n", i, num_array[i]);
    }
    n = 4;
    num_array = realloc(num_array, n*sizeof(int));
    num_array[3] = 1500;
    for (int i = 0; i < n; i++){
        printf("i: %d. arr[i]: %d\n", i, num_array[i]);
    }
    free(num_array);
}
```

Pointers + functions

- Which of the following change the value of n or m?
- Why?

```
n: 3, m: 30, ptrval: 3
n: 3, m: 30, ptrval: 3
n: 3, m: 30, ptrval: 3
n: 4, m: 30, ptrval: 4
n: 4, m: 31, ptrval: 4
```

```c
void inc_1(int s){
    s += 1;
}

void inc_2(int * s){
    *s += 1;
}

int main(int argc, char *argv[]) {
    int n = 3, m = 30;
    int * ptr = &n;
    printf("n: %d, m: %d, ptrval: %d\n", n, m, *ptr);
    inc_1(n);
    printf("n: %d, m: %d, ptrval: %d\n", n, m, *ptr);
    inc_1(*ptr);
    printf("n: %d, m: %d, ptrval: %d\n", n, m, *ptr);
    inc_2(ptr);
    printf("n: %d, m: %d, ptrval: %d\n", n, m, *ptr);
    inc_2(&m);
    printf("n: %d, m: %d, ptrval: %d\n", n, m, *ptr);
}
```

Read/write I/O

- getchar(), putchar(), gets(), puts(), EOF
- get/putchar() - one character at a time
  - Useful for parsing input character by character
  - EOF = end of file

```
int main(int argc, char *argv[]) {
    char c;
    while ( (c = getchar()) != EOF){
        if (c < 'g'){
            putchar(c);
        }
    }
}
```

Files

- Instead of stdin, stdout, stderr, use our own files
- Example:
  - Read from one file, output to another
  - Character by character
- Remember to close afterwards
- fopen(filename, permissions)

```c
int main(int argc, char *argv[]) {
    FILE *in_fp, *out_fp;
    in_fp = fopen("input.txt", "r");
    out_fp = fopen("output.txt", "w");
    char c;
    while ( (c = fgetc(in_fp)) != EOF){
        fputc(c, out_fp);
    }
    fclose(in_fp);
    fclose(out_fp);
} /* */
```

Qsort

- qsort example
  - Pass in pointer to function, uses this fn to compare values
- qsort(arr, len(arr), element size, compare fcn)
- What if we swap a, b?

```c
int compare_fn(const void *a, const void *b)
{
    return (*(int*)b) - *(int*)a;
}

/* qsort ex */
int main(int argc, char *argv[]) {
    int arr[] = {1, 2, 5, 90,10, 421, 0};
    qsort(arr, 7, sizeof(int), compare_fn);
    for (int i=0; i < 7; i++){
        printf("index: %d, sorted value: %d\n", i, arr[i])
    }
}
```

Questions??