CS241 - Intro to C

This week we are going in depth about strings, arrays, and structs, more on the first two.

Pointer Arithmetic

When you add x to a pointer, C takes the base type of the pointer and moves the pointer `sizeof(basetype)*x` bytes.

Given the following addresses:

```
int *int_array = (int *)0x100;
char **string_array = (char **)0x200;
string_array[0]  == (char *)0x300;
string_array[1] == (char *)0x400;
sizeof(int) == 4
sizeof(char*) == 8
sizeof(char) == 1 //but you knew that...
```

Write out the following addresses:

- `int_array + 3 ==0x`
  `string_array + 2 ==0x`
  `string_array[0] + 5 ==0x`
  `string_array[2] + 10 ==0x`

What does the memory look like?

This exercise is to help you build a mental model of what the memory looks like. Draw out the memory in boxes. For example

```
char *example = malloc(6); //Address 0xF00
strcpy(example, "camel");
```

example =

c	a	m	e	l	\0

(There is no hard and fast rule about the format, whatever makes the most sense to you).

```
    char** first_name = malloc(3*sizeof(char *));
        //Address 0x100
```

- `first_name =`

  `first_name[0] =`

  `first_name[1] =`

```
    first_name[0] = malloc(7); //Address 0x200
    first_name[1] = NULL;
```

- `first_name =`

  `first_name[0] =`

  `first_name[1] =`

```
    strcpy(first_name[0], "bhuvan");
```

- `first_name =`

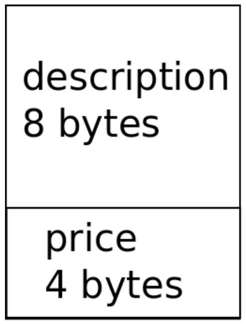
  `first_name[0] =`

  `first_name[1] =`

A bit about structs

In this section we will give you a struct and you'll have to draw it out in memory

```
struct product{
    char* description;
    float price;
};
```



```
struct shelf{
    product items[2];
    char* description;
};
```

```
struct aisle{
    shelf *shelves;
    size_t num_shelves;
};
```

```
struct store{
    aisle *aisles;
    size_t num_aisles;
    int store_code;
    char *name;
    char *description;
};
```