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 \( \text{sizeof(basetype)} \times x \) bytes.

Given the following addresses:

- \( \text{int} *\text{int\_array} = (\text{int} *)0x100; \)
- \( \text{char **string\_array} = (\text{char **})0x200; \)
- \( \text{string\_array[0]} == (\text{char} *)0x300; \)
- \( \text{string\_array[1]} == (\text{char} *)0x400; \)
- \( \text{sizeof(int)} == 4 \)
- \( \text{sizeof(char*)} == 8 \)
- \( \text{sizeof(char)} == 1 //but you knew that...} \)

Write out the following addresses:

- \( \text{int\_array + 3} == 0x \)
- \( \text{string\_array + 2} == 0x \)
- \( \text{string\_array[0]} + 5 == 0x \)
- \( \text{string\_array[2]} + 10 == 0x \)

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

- \( \text{char *example = malloc(6); //Address 0xF00} \)
- \( \text{strcpy(example, "camel"}); \)
- \( \text{example} = \text{c a m e l} \backslash 0 \)

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

- \( \text{char** first\_name = malloc(3*\text{sizeof(char*)});} \)
  
  //Address 0x100

  - \( \text{first\_name} = \)
  
  - \( \text{first\_name[0]} = \)
  
  - \( \text{first\_name[1]} = \)
  
  - \( \text{first\_name[0]} = \text{malloc(7); //Address 0x200} \)
  
  - \( \text{first\_name[1]} = \text{NULL;} \)

  - \( \text{first\_name} = \)
  
  - \( \text{first\_name[0]} = \)
  
  - \( \text{first\_name[1]} = \)
  
  - \( \text{strcpy(first\_name[0], "bhuvan");} \)
  
  - \( \text{first\_name} = \)
  
  - \( \text{first\_name[0]} = \)
  
  - \( \text{first\_name[1]} = \)

- \( \text{char *first\_name = malloc(7); //Address 0x200} \)
  
  - \( \text{first\_name[0]} = \)
  
  - \( \text{first\_name[1]} = \)
  
  - \( \text{strcpy(first\_name[0], "bhuvan");} \)
  
  - \( \text{first\_name} = \)
  
  - \( \text{first\_name[0]} = \)
  
  - \( \text{first\_name[1]} = \)
A bit about structs

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

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

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

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

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