MPP - Structs Exercise

Dominic Carr

1 What is a Struct?

As discussed in the previous videos a struct, short for **structure**, is essentially a composite data type. You can define what kinds of variable types exist within a struct. Like methods they are designed to provide **structure** to a program. They can group pieces of data together which “belong” with each other. For example in Listing 1 we see a struct representing a person with a name, a salary, and government id number.

```c
struct Person {
  char name[50];
  int govID;
  float salary;
};
```

Listing 1: Simple Struct in C

2 Tasks

Create structs to represent...

- An employee (name, age, salary, years worked, job title)
- A college module (module name, credits it is worth)
- Student (name, age, an array of their college modules)
- Manager (name, age, salary, an array of the employees they manage)

The student structure should have internally an array of college modules. The same goes for the manager and their employees. In Listing 2 we have a partial solution for some of the above tasks. We have a Student structure which holds an array of 10 modules, this means that each student we create can have 10 modules. We add two modules to the student struct and we print them out. You can expand on this to add all the other information specified above such as name and age. It would be useful to keep track of how many modules have been added to each student, so that we know what index to place the next one. This can be done by storing an integer in the struct to keep track.

For each struct type create a print method which outputs to the console all the information stored in the struct.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Module {
  char name[50];
};

struct Student {
  struct Module modules[10];
};

void printModule(struct Module module){
  printf("the module name is %s\n", module.name);
}

int main(void)
{
  struct Module module = { "Multi-paradigm Programming" };
  printModule(module);

  struct Student student = {};
  student.modules[0] = module;

  struct Module module2 = { "Introduction to Programming" };
  student.modules[1] = module2;

  printModule(student.modules[0]);
  printModule(student.modules[1]);


    return 0;
}
```

Listing 2: Struct Example