CS5008 - LAB 4

The Lab 4 task will help us understand programmer designed data types in C. Specifically, we are going to create a VECTOR type that will store data and that we shall create functions to manipulate the data

References:
https://cplusplus.com/reference/cstdio/printf/
https://cplusplus.com/reference/cstdio/scanf/
https://www.simplilearn.com/tutorials/c-tutorial/structure-in-c

The learning outcomes (**What Prof Mwaura hopes that you will learn**) for this Lab are:

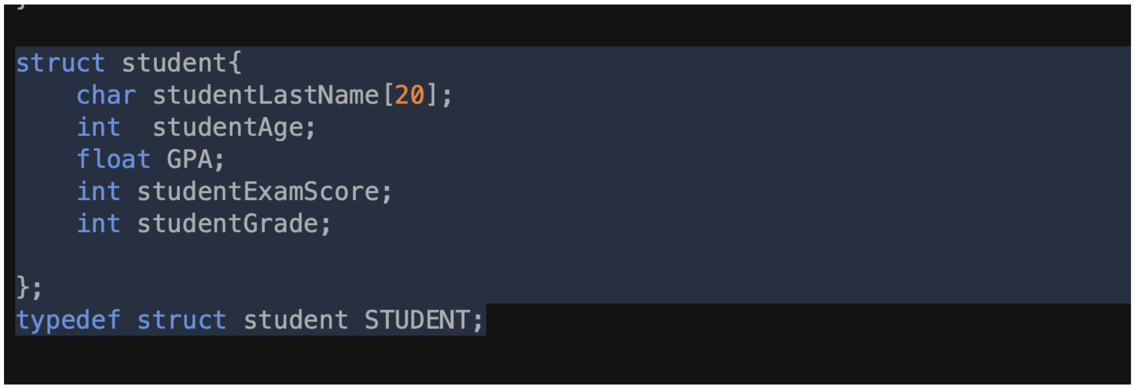
- Conversant with user defined types in C
- Uses Enums and typedef
- Conversant with Structs in C
- Ability to use C structs to create a data structure.
- Uses a Vector type to store, and retrieve data
- Uses dynamic memory allocation to allocate memory.

Task 1 (10 points):

The work in this task should be carried out in your lab04.c
This section assumes that you have completed Lab 2. In your Lab 2 you created a program that allows for collection of the following data:

1) Student Last Name
2) Student Age
3) Student GPA
4) Student Final Exam Score
5) Student Grade - This was computed based on Student Final Exam score.

You created a struct type called student and you used the typedef keyword to create a synonym for your type and called this STUDENT. Your code may have looked like this:


```
struct student{
    char studentLastName[20];
    int  studentAge;
    float GPA;
    int studentExamScore;
    int studentGrade;

};
typedef struct student STUDENT;
```

An important aspect that we have observed regarding arrays is that we need three main parameters to work with arrays. These are:

- An initial capacity of the array
- A count of how many items we have in the array
- A name of the array that acts as a pointer or storage for a memory of the first element.

Using these attributes we can maneuver various operations in an array and using dynamic memory management we can create dynamic arrays.

For this lab, Prof Mwaura is interested in keeping data containing only student exam scores. Essentially, he only needs an array that keeps track of the scores and uses the array index as the student ID. Recall that the count in terms of number of elements can be used as the indices of the array.

Task: Modify your struct to now only keep the following items:

- Capacity of the array (call this capacity)
- A count that keeps track of the total number of items in the array (Call this total)
- A pointer variable that keeps the address (call this data).

Note that you can delete all other member variables you had before as they are not required anymore.

Task 2 (20 points):

a) In your main function, you will now need to create a variable of STUDENT type to test whether your struct type works. Set initial capacity as 7, note that you will need to use dynamic memory allocation to create your array, making sure that you initialize all values in the array as zeros.

b) Using a looping mechanism of your choice, use the total member variable and the capacity member variables to populate the array. Note that you can use the rand function and have values between 0 and 100 stored in the array.

c) Create a function called printData that will receive only one argument, the STUDENT type variable and shall use the total member variable and the data array to print out all the items in the array. The function signature will look like this:

```
void printData(STUDENT myStudent);
```

d) Ensure that you free the memory used with the data array.

Checkpoint 1: Show the TA or the professor your work and demonstrate that your code works with no errors or memory leaks.

Task 3 (10 points):
Create a function that can be used to push or insert data into the array. Your function should only receive the STUDENT struct variable. The function signature should look like this:

```
 void pushData(STUDENT myStudent);
```

For your function definition, just copy and paste the for loop from your main and then call the function. Note that this function should be a non value returning function.

Q1. Does your function work, precisely, are you still able to use your print function to print items in the array? (2 points)

Q2. If your function does not work (you do not seem to print anything)? Please provide an explanation why it does not work and what you could do to make it work (7 points). Note that you can also test inside the loop to see whether data is being added into the array.

Task 4 (25 points)

In this task, you will explore working with pointers to structs. Recall that pointers are special variables that store addresses. For instance, in tasks 1 and 2, you used a pointer variable to store the address of the first element in the array. To declare a pointer variable, you used the following syntax:

```
int * data;
// int is the data type of the variable whose address data is kept. The
* is a reference operator when used in this manner.
```

Similar to any other type, we can use pointers to variables made of the struct type. In this section we shall do the following:

a) Create a pointer variable that stores the address of the struct variable. Note that for this case, we are just creating a pointer and then assigning the address of the struct variable that we had before.
b) Modify your printing function to now accept a pointer to struct variable. The modified function signature should look like this:

```
void printData(STUDENT* myStudent);
```

How would you use the pointer variable to access the underlying member variables? Recall that in the past use of pointers, we used a dereference variable to do so. For instance, consider

```
int a = 8;
int * pA = &a;

//The following statement dereferences the pointer and assigns value 9.
The variable a now has 9
*(pA) = 9;
```

We could achieve the same using our pointer to struct variable. Remember that it's the variable to the struct that we have a pointer for and not the members. In that case, we can just dereference the variable and use a dot member operator to access the member variable.
E.g

```
STUDENT goodStudent; //create a variable

//create a pointer variable and assign
STUDENT * pgoodStudent = &goodStudent;

/*
use the pointer for assignment. Note the parentheses showing I am
dereferencing the pointer variable only and note the entire pointer and
member variable.
Note I am using the * with struct to pointer and then using the member
dot operator to access the member variables.
*/
 (*pgoodStudent).capacity = 10;

Another way to achieve exactly the same thing is to use the pointer to
struct operator (also called an arrow operator). We do it this way:
(please note that this does not use the member dot operator).

pgoodStudent->Capacity = 10;
```

c) Now modify your earlier created pushData function to use the pointer to struct variable. This signature should look as follows:

```
void pushData(STUDENT * myStudent);
```

Please report on your observations, does your function work? Precisely, are you able to now use it to update the array?

Task 5 (5 points)
Variables (whether made from structs or other data types) that are created in main, use the stack memory. This is not always the best thing to do in particular if your data might grow. For instance, in your current work, the data array has its memory in the Heap and your struct's memory is in the stack.

In order to modify this, you would need to have the struct memory assigned using dynamic memory management. You can use malloc to do this.

Your program already has a pointer to struct, the only thing to do is assign the memory using malloc instead of using the pointer to struct variable.

NOTE: if you have done it properly, your entire program will now show to memory allocations. The first one is for the struct and the second one is for the array.

Remember you will need to free both memory items. Please think deeply about how to free that.

Task 6 (30 points)

Task 5 has moved us from stack frame memory to heap memory. Lets extend this work by creating two new functions as follows and also modifying the PushData function .

```
/*
This function will not receive any arguments but will return a pointer
to STUDENT. The function will be used to do the memory allocation for a
struct variable (using a pointer) and will initialize all member
variables including allocating memory for the array.
Essentially, copy your code that is doing initialization from main and
place it in this function. The returned pointer will be assigned to the
declared pointer to struct in main.
Make sure to test your code:
This here works like your constructor function when you did python.
*/
```

```
STUDENT* initialize(void);

/*
This function will receive the address of the pointer to struct from
main. It will then dereference the pointer and use the obtained pointer
to release all allocated memory.
Make sure you assign to NULL after everything has been freed.
This is a deconstructor.
*/
void  deallocate(STUDENT ** myStudent);

/*
Finally, modify your pushData so that it accepts one item at a time.
Your function should also allow for updating of your capacity to be
able to increase the capacity of the array whenever we need to.
Please refer to the vector class if you have any questions regarding
how to do this.
*/
void pushData(STUDENT * myStudent, int item);
```

Final checkpoint: Demonstrate to TA and professor that you code works as expected and does not have any memory leaks.

Make sure that you push this to github and that your tested this with the Khoury Server.