BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)
CS F111 Computer Programming
LAB SESSION #9
(Structures in C)

Exercise - Structures in C:

Now having learned about multi-file compilation and creation of a C project, let us now create a C project related to structures in C. Consider a library. Every book (say) is associated with the following fields: **ID** (integer), **shelfNum** (integer), **price** (float). We want to maintain a catalog of books (with the above fields) that are there in the library. We have to use structures for this. Please refer to the link- https://drive.google.com/file/d/1X_aDHvBj9VgXlx-qZY2ANu-pxqjsFm0N/view?usp=sharing for the incomplete code implemented in a modular way. It has 5 files in it: **"book_def.h"**, **"book_fun.c"**, **"books_catalog.c"**, **"books_catalog.h"**, **"main_library.c"**. The structure definitions that are related to defining a book are present in **"book_def.h"**. It also has a few functions which you should implement in **"book_fun.c"** as per the specifications given in **"book_def.h"**. Similarly, **"books_catalog.h"** defines a global array **booksCatalog** of the type **struct book** (typdefed as BOOK) in order to store books. **MAX_SIZE** has been used to give a maximum size to that array. The variable **count** is used to keep a track of number of books present in the **booksCatalog** array. Remember how we added and deleted objects from an array by keeping a max size array in the lecture hours. You should use that concept over here to implement this. There are some functions declared in **"books_catalog.h"**, which you should define in **"books_catalog.c"**. The **main()** function is implemented in **"main_library.c"**. This file is complete and you don't have to make any changes to it.

What you will have to do:

- Implement all functions in **"book_fun.c"** and **"books_catalog.h"** as per the specifications given in header file along with the function declaration.
- Create a **"myScript.sh"** file that compiles and executes the above files. Remember to include commands to remove previously compiled **".o"** file and the executable.
- Notice that the variable count has been declared in **"book_catalog.h"**. However no value has been assigned to it in that file. The **main()** function in **"main_library.c"** assigns a value to it when the program starts. This is a good practice to avoid run-time errors. It is a good practice to always assign values to global variables within some functions and not in the global space of the **".c"** file or inside a **header file**. You should only declare them in the global space or the header file and not assign a value to them.
- Carefully see and study the file inclusions used. Wherever (in any **".c"** file) we want to use a global variable or define a function that is present in a given header file, we should include the header file in our **".c"** file. A header file can be included in multiple **".c"** files. Just like **"book_def.h"** has been included in **"book_fun.c"**, **"book_catalog.h"**, and **"main_library.c"**.
- Please notice that there are certain guards placed in every .h file using **#ifndef**. This is essentially used to solve the error that comes by multiple inclusions of a single header file. For example, the header file **"book_def.h"** is included in **"book_fun.c", "book_catalog.h"** and **"main_library.c".** If you try to compile the entire project without these guards, you may end up getting a compilation error. These guards prevent such errors by making sure that the compiler processes each header file only once. The standard way of using these guards is illustrated by the following example and syntax:

```
// myheader.h

#ifndef HEADER_FILE
#define HEADER_FILE

// the entire header file file

#endif
```

The word **HEADER_FILE** is just a label given to our current **"myheader.h"** which is the actual **".h"** file. The above construct is commonly known as a wrapper **#ifndef**. It means: **"if not defined"**. It says that if the label **HEADER_FILE** is not defined (i.e. if we have not yet compiled any **".c"** file that has included **"myheader.h"** in it), then define it (compile it) now. If the label **HEADER_FILE** has already been defined (by compilation of some other ".c" file that includes **"myheader.h"**), then don't define it again. In this way each ".h" file is processed by the compiler only once and prevents possible compile time error. It is left as an exercise for you try the following:

- Remove all the guards and then compile the entire project. You should encounter a compile time error.

Additional Practice Exercises

Q1. In this exercise, you need to model a marks database for storing various students' marks and then perform various operations on them. You need to write a `struct student` in order to implement this. Every student record has a **ID**(integer), **Name**(char array – you can assume each name is at max of 25 characters), **Marks**(integer array) and **Avg**(float). The **Marks** array is an array of 5 integers which denote the marks obtained by a student in 5 different courses.
Your program should ask the user for the number of students, followed by their ID, name and marks in the 5 courses. Then your program should calculate the average for each student and store them in each student record. Finally display all the information for all the students.
You are expected to write a proper modularised code with separate functions for each task. Feel free to use multiple files to structure your code efficiently.