Advanced Programming – PROG36859

Workshop # 2

Arrays, C strings (null-terminated array of characters) and functions

In the process of doing your workshop 2, you will create a **Subject Mark Report** program. This program will receive Subject-Code (i.e., PROG36859) and a series of student information (student number and mark) and print a report based on the received information.

Learning Outcomes

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- Use Cstrings to hold text.
- Use arrays to hold the formation to process later.
- Pass read-only arrays to functions.
- Pass arrays to functions and fill them with information remotely (using the function).
- Describe to your instructor what you have learned in completing this workshop.

Submission Policy

This workshop is divided into two coding parts and one non-coding part:

- Part 1: A step-by-step guided workshop, worth 20% of the workshop's total mark

- Part 2: A Do It Yourself type of workshop that is much more open-ended and is worth 50% of the workshop's total mark.
- *reflection*: non-coding part, to be submitted together with *DIY* part. The reflection is worth **30%** of the whole workshop's mark. If your professor deems it insufficient, you will lose 30% of your workshop mark.

Due Dates

The Due dates depend on your section. You can check the Slate for the due dates. You have approximately 1 week to complete all the parts of the workshop and submit your solution to the slate. Follow the instructions of the submission at end.

Late penalties

You are allowed to submit your work up to 3 days after the due date with 10% penalty for each day. After that, the submission will be closed, and the mark will be zero.

Citation

Every file that you submit must contain (as a comment) at the top:
your name, **your Sheridan email**, **Sheridan Student ID** and the **date** when you completed the work.

For work that is done entirely by you (ONLY YOU)

If the file contains only your work or the work provided to you by your professor, add the following message as a comment at the top of the file:

I have done all the coding by myself and only copied the code that my professor provided to complete my workshops and assignments.

For work that is done partially by you.

If the file contains work that is not yours (you found it online or somebody provided it to you), **write exactly which part of the assignment is given to you as help, who gave it to you, or which source you received it from.** By doing this you will only lose the mark for the parts you got help for, and the person helping you will be clear of any wrongdoing.

- Add the citation to the file in which you have the borrowed code.
- In the 'reflect.txt` submission, add exactly what is added to which file and from where (or whom).

: warning: *Workshops with no Citation will receive a mark of zero.*

: warning: *This Submission Policy only applies to the workshops. All other assessments in this subject have their own submission policies.*

If you have helped someone with your code

If you have helped someone with your code. Let them know of these regulations and in your ‘reflect.txt’, write exactly which part of your code was copied and who was the recipient of this code.
By doing this, you will be clear of any wrongdoing if the recipient of the code does not honour these regulations.

Compiling and Testing Your Program

All your code should be compiled and check the output, and closely compare it with the expected output.

Part – 1 (20%)

Subject Marks Report Program

printReport

```
void printReport(const char subjectCode[], const int studentNumbers[],
                                        const int marks[], int noOfStudents)
```

In this part of the workshop, code a function to generate a report from information stored in several arrays in a module called `classList` (i.e., classList.h and classList.c)

The tester program printReportTester.c demonstrates how the function is used and what is the output:

```
#include "classList.h"
int main() {
   int stno[4] = { 12345, 23456, 345678, 45678 };
   int mark[4] = { 55,66,44,88 };
   printReport("PROG36859", stno, mark, 4);
   return 0;
}
```

printReport() output:

```
SUBJECT MARKS REPORT!
  +---------------+
  | PROG36859     |
  +--------+------+
  | Std No | mark |
  +--------+------+
  | 012345 |   55 |
  | 023456 |   66 |
  | 345678 |   44 |
  | 045678 |   88 |
  +--------+------+
  |Average |   63 |
  |Highest |   88 |
  |Lowest  |   44 |
  +--------+------+
```

implementation

step 1

Add the function body to `classList.c` and the prototype to `classList.h`. In your function, you will add four integer variables. Specifically, you will be adding a variable for your loop counter, the average mark, lowest mark, and highest mark.

step 2

Make the function only print the first six lines of output. Use the output sample of the static lines and use `%s` format specifier in printf to print the subject Code. Use a specific width setting in `%s`

to have the surrounding bars (|) aligned. To review how to align our character string's output, check out below:

Qualifiers

Qualifiers on the `%s` specifier add detail control:

- `%20s` displays a string **right**-justified in a field of **20**
- `%-20s` displays a string **left**-justified in a field of **20**
- `%20.10s` displays the ***first*** **10** characters of a string **right**-justified in a field of **20**
- `%-20.10s` displays the ***first*** **10** characters of a string **left**-justified in a field of **20**

Now test the function and make sure you get this output:

```
SUBJECT MARKS REPORT!
  +---------------+
  | PROG36859     |
  +--------+------+
  | Std No | mark |
  +--------+------+
```

step 3

In a loop that starts from zero that goes up to the `number of students` argument value - print each data line with student number and marks. These values are held in the elements of the `student numbers` array argument and the `marks` array argument

step 4

While looping, keep track of the lowest and highest marks (possibly by checking the values) and the value of each `marks` element to the average mark.

step 5

Divide the value of the average by the number of students and then print the last five lines of the report using these values.

main.c

Add the code in the main.c file as commented in the file (main.c is provided).

Messages and data entry formats:

Student Information Entry Prompt

```
Enter ## student numbers and student marks...
```

is to be replaced with the number of students

Student Data Entry Format

```
RR:
Student Number: 123456
Mark: 99
```

- RR is replaced with the row number of the current data entry. (1 to number of students)
- 123456 is an example of a valid user-entered student number
- 99 is an example of a valid user-entered student mark

execution sample:

```
Enter subject Name: PROG36859
Enter the number of students (max 50): 4
Enter 4 student numbers and student marks...
1:
Student Number: 11111
Mark: 11
2:
Student Number: 22222
Mark: 22
3:
Student Number: 33333
Mark: 33
4:
Student Number: 44444
Mark: 44

SUBJECT MARKS REPORT!
  +---------------+
  | PROG36859     |
  +--------+------+
  | Std No | mark |
  +--------+------+
  | 011111 |   11 |
  | 022222 |   22 |
  | 033333 |   33 |
  | 044444 |   44 |
  +--------+------+
  |Average |   27 |
  |Highest |   44 |
  |Lowest  |   11 |
```

```
+--------+------+
```

Part 2 DIY (50%)

Improve your lab work by making your number of data entries foolproof and your report repeatable using the provided main function.

Code improvement

step 1

Study the main function below. You will be coding `void subjectMarksReport(void)` and `int getMMInt(int min, int max, const char valueName[])`.
For `void subjectMarksReport(void)`, put its definition in `classList.c` and its prototype in `classList.h`.
For `int getMMInt(int min, int max, const char valueName[])`, put its definition in `utils.c` and its prototype in `utils.h`.
Note, `void subjectMarksReport(void)` call the `int getMMInt(int min, int max, const char valueName[])` that you code.

step 2

compile and run your code with the provided main tester and make sure everything works.

```
#include <stdio.h>
#include "classList.h"
#include "utils.h"
int main(void) {
   int i;
   printf("Subject Performance report\n");
   printf("-------------------------\n");
   do {
      subjectMarksReport();
      printf(" Exit? ");
   } while(!yes());
   return 0;
}
```

step 3

```
int getMMInt(int min, int max, const char valueName[]);
```

add a function to utils.c to get an integer within a range. Call the `getInt()` function for integer validation and then check the received value to be between min and max. If not print the following message:

```
[1<=??????<=50], try again: _
```

replace ??? with the valueName received from the argument list.

For example the following function call:

```
int age:
printf("Enter your age: ")
age = getMMInt(18, 99, "AGE");
printf("You are %d years old!\n", age);
```

should run as follows:

```
Enter your age: 9
[18<=AGE<=99], try again: abc
Invalid Integer, try again: 100
[18<=AGE<=99], try again: 40
You are 40 years old!
```

step 4

Have all your integer entries use the `getMMInt` function with the following value names and ranges:

```
[1<=Number of students<=50]
[9999<=Student Number<=1000000]
[0<=Mark<=100]
```

step 5

Improve your report output as displayed below:

```
 SUBJECT MARKS REPORT!
 +--------------------+
 | PROG36859          |
 +---+--------+------+
 |Row| Std No | mark |
 +---+--------+------+
 | 1 | 012345 |   66 |
 | 2 | 023456 |   77 |
 | 3 | 034567 |   88 |
 | 4 | 045678 |   99 |
 +---+--------+------+
 | Average    |   82 |
 | Highest    |   99 |
 | Lowest     |   66 |
 +------------+------+
```

Reflection (30%)

Study your final solutions for each deliverable of the workshop, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. **This should take no less than 30 minutes of your time and the result is suggested to be at least 150 words in length.**

Create a file named `reflect.txt` that contains your detailed description of the topics that you have learned in completing this workshop and mention any issues that caused you difficulty.

You may be asked to talk about your reflection (as a presentation) in class.

Submission Process:

Data Entry for Part 2

```
PROG36859
5
1111111
11111
abc
101
100
22222
22
33333
33
44444
44
55555
55
```

Follow the instructions during submission

Files to Submit

```
main.c
classList.h
classList.c
utils.h
utils.c
```

Submission to Slate

Zip the whole visual studio project and upload it to slate.

- Your VS solution should be having two projects.
- For part 1 and part 2