CS200 Computational Problem Solving
Spring 2020-21

Monday 25 January 2021

Lab 02: Exercise
Practice Lab (Ungraded)

Lab Guidelines

Please note that this lab is not graded. It is just a practice exercise based on the CS100 course content.

1. Make sure you get your work graded before the lab time ends.
2. You put all your work into the folder **Lab01_YourRollNo_TAname** and submit it on LMS (Assignment>Lab01) before the time the lab ends.
3. Talking to each other is NOT permitted. If you have a question, ask the lab assistants.
4. The object is not simply to get the job done, but to get it done in the way that is asked for in the lab.
5. Phone is NOT allowed. Put it in bag or at instructor desk.
6. Any cheating case will be reported to Disciplinary Committee without any delay.

Coding Conventions:

1. Constants are “ALL_CAPS”.
2. Variables are “all_small”.
3. All function names must be “firstWordSmallAllOtherWordsCamelCase”.
4. All curly brackets defining a block must be vertically aligned.

Learning Objective:

1. PO-02 Develop proficiency in the practice of computing.
2. CO-02 To help students analyze and solve programming problems
3. LO-02 Critical Thinking and Analysis
4. LO-03 Problem Solving
5. LO-05 Responsibility

Marks: Name: ______________________ Roll #: __________________

Task0									**Total**
									10

Task1									**Total**
									20

Task 2									**Total**
									30

Task 3									**Total**
									40

Total Marks Obtained

/100

TA: _______

Let’s Begin

Task 0: [10 Marks]

- This task is to give you the best of luck for the rest of the semester.
- Ask the user to input their name as a string. Then, just print out “Best of Luck name”.
- For example, if the user enters “Hamza” as their name, your program should print out “Best of Luck Hamza”.
- Run the program, enter your name, and wish yourself best of luck.

Task 1: [20 marks]

- This task will help you get proficient at taking user input.
- You will create a program that calculates the sum of some integers. How many integers? That depends on what the user wants.
- Start the program by asking the user how many integers he/she wants to add. Based on this number, take integer input from the user. So, for example, if the user wants to add 5 numbers, you should take 5 integers as input from the user.
- Once the input is taken, display the sum of all the integers in the console.

Task 2: [30 marks]

- This task will help you master pointers, functions, and arrays.
- Copy the code for the above task into a new .cpp file and modify that code as follows.
- The program starts in the same way, i.e. by asking the user how many integers he/she wants to enter. Then, take that many integers from the user as input and store them in an array.
- Now, create a function called findMinMax() and pass the array of integers to the function (you’ll pass pointer to the array).
- This function does two things:
    - It finds the smallest integer in the array (minimum).
    - It finds the largest integer in the array (maximum).
- Once the minimum and maximum have been found, create a new array called result, and store the minimum and maximum in the array. Now return this array. You will again have to return a pointer to this array.
- In the main function, catch the result array that is being returned from the function and then display the minimum and maximum in the console. For example,

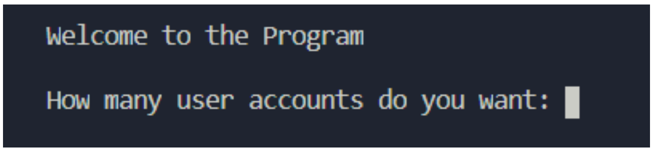
```
Minimum = 1
Maximum = 100
```

Note:

- You are not allowed to display the Minimum and Maximum from inside the findMinMax() function. It must be displayed from the main function.
- Assume that the input integer array will not have less than 2 integers.
- Assume that the input will be valid integers.

Task 3: [40 marks]

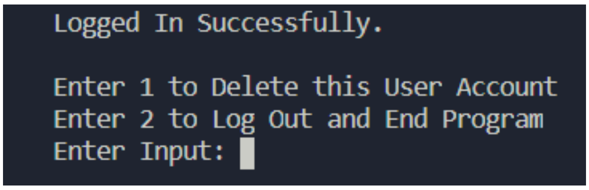
- In this task you will create a basic Login and Signup system. You will store the usernames and passwords of the users in arrays as strings. There will be one array called usernames which will store all the usernames, and another array storing passwords.
- The initial size of the username and password arrays should be decided by the user and must be greater than or equal to 3. The array must be **dynamically allocated** so that we can increase its size if more users come.



```
Welcome to the Program

How many user accounts do you want: █
```

- Now, ask the user to enter the usernames and passwords. For example, if the user decided that the size of the arrays would be 3, then ask the user to input three usernames and three passwords and store them in the respective arrays.
- Once the arrays have been filled, ask the user to log in by entering a username and a password. Verify that the username and password are correct. If not, ask again and keep asking until a valid set of username and password is not entered. Once the user enters the correct username and password, he/she is logged in and a message “Logged In Successfully” is displayed.
- Now, the following interface is shown to the user:



```
Logged In Successfully.

Enter 1 to Delete this User Account
Enter 2 to Log Out and End Program
Enter Input: █
```

- Based on the user’s input, if the user inputs 1, delete that user account from the username and password arrays. If the user enters 2, simply end the program.
- Note: You can assume that the user enters valid usernames and passwords.