NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD

Programming Fundamentals (CS118)

FALL 2019 ASSIGNMENT # 4

Due Date: Monday, November 18, 2019 (1:00 pm)

Total Marks: 130 **Bonus Marks: 20**

Instructions

Submission: Combine all your work (solution folder) in one .zip file. Use proper naming convention for your submission file. Name the .zip file as **SECTION_ROLL-NUM_04.zip** (**e.g. F_19i0412_04.zip**). Submit .zip file on Google Classroom within the deadline. Failure to submit according to the above format would result in deduction of 10% marks. Submissions on the email will not be accepted.

Plagiarism: Plagiarism cases will be dealt with strictly. If found plagiarized, both the involved parties will be awarded zero marks in this assignment, all of the remaining assignments, or even an **F grade** in the course. Copying from the internet is the easiest way to get caught!

Deadline: The deadline to submit the assignment is **18th November 2019 at 1:00 PM**. Late submission with marks deduction will be accepted according to the course policy shared earlier. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Comments: Comment your code properly. Bonus marks (maximum 10%) will be awarded to well-commented code. Write your name and roll number (as a block comment) at the beginning of the solution to each problem.

Tip: For timely completion of the assignment, start as early as possible. Furthermore, work smartly - as some of the problems can be solved using smarter logic.

Note: Follow the given instructions to the letter, failing to do so will result in a zero.

Problem 1: Numeric input validation using strings **[10 Marks]**

There are various problems with input validation, especially when reading numeric data types using **cin**. Although, cin.fail() gives us some sort of solution but still it cannot handle all scenarios, for example, 123abc is considered a valid input because 123 is read where as it an alphanumeric input.

You are required to write a program that read input as a character array/string and validate it. Table 1 shows valid and invalid input for different numeric data types.

Table 1: Numeric data types

Date type	Valid input	Invalid input
Integers	//Only digits 1 2132 -3121	123abc 123.123 aqwe121 _12 +121
Float	12.23f -12.23f	123abc 123.123
Double	12.23 -12.23	123abc 123.123asdasda 12.2f

Note: *All the invalid values which cin.fail() considers should also be invalid for the program that you write.*

Problem 2: Sorting and Merging **[10 Marks]**

Write a program that takes 5 integer arrays (*of varying sizes*) as input. You have to ensure that user enter these arrays in ascending order, if user enters incorrectly display a prompt to read input in correct format. Write a C++ program to produce an array that merges elements of all arrays in descending order, but it also needs to remove duplicates.

Example:

Array A_1: {1, 3, 5, 6}
Array A_2: {1, 2, 4, 8}
Array A_3: {5, 6, 7, 8}
Array A_4: {23, 24, 94, 108}
Array A_5: {1, 2, 2, 23, 24, 67, 1234}

Merged array: {1234, 108, 94, 67, 24, 23, 8, 7, 6, 5, 4, 3, 2, 1}

Note: Final array should be created/merged in sorted order. You are not allowed to do any computation on the elements entered in the Merged array. This means that different numbers will be inserted in this array in descending order and duplication will be checked at the time of insertion. Also, no temporary arrays are allowed.

Problem 3: Employee performance reporting **[20 Marks]**

Write a program that calculates performance based salary of customer support representatives (CSRs) in an organization:

1. csrID (Customer support representative ID): an array of seven strings to hold employee identification numbers. The array should be initialized with the following numbers:
   CSR_01, CSR_02, CSR_03, CSR_04, CSR_05, CSR_06, CSR_07
2. csrName: an array to hold the name of each CSR, input at run time
3. hours: number of hours worked by each CSR, input at run time
4. complaintsResolved: number of complaints successfully resolved by each CSR, input at run time
5. payRate: an array of seven to hold each employee s hourly pay rate, where payRate is calculated as following:
   payRate = $25 + 25*(complaintsResolved by each CSR/total complaints resolved)
6. wages : an array hold each employee's wages
   wages = hours * payRate

The program should relate the data in each array through the subscripts. For example, the number in element 0 of the hours array should be the number of hours worked by the CSR whose identification number is stored in element 0 of the csrID array. The program should do the following:

1. Display each csrID and ask the user to enter names, hours and complaintsResolved.
2. It should then calculate the payRate and gross wages for that CSR and store them in the payRate and wages array, respectively.
3. After the data has been entered for all the CSRs, the program should display each CSR's identification number, name and gross wages.
4. Program should also display options to display the top **N** CSRs on the basis of different criteria, including: number of complaintsResolved, number of hours worked.

Note: *Ensure all necessary input validation, for example, hours, complaintsResolved cannot be negative numbers.*

Problem 4: Student record **[40 Marks]**

Write a program that helps a faculty member prepare his semester result and calculate grades of the students. Table 2 show a typical data that a faculty member would have towards the end of the semester for any course.

Table 2: Course data

Reg No.	*Midterm (25)*	*QUIZZES (10)*										*Assignments (15)*					*Final Exam (50)*	*Final Score*	*Grade*
		Q1	*Q2*	*Q3*	*Q4*	*Q5*	*Q6*	*Q7*	*Q8*	*Q9*	*Total*	*A1*	*A2*	*A3*	*A4*	*Total*			
	13	8	6	6	6	5	3	3	0	0	6	10	7	7	0			45	F
	17	8	7	7	6	5	3	3	0	0	6	10	7	7	0			70	B
	11	9	8	5	4	3	3	0	0	0	5	10	7	7	0			86	A
	11	8	5	4	1	0	0	0	0		3	10	7	7	0			60	C

Write a program that can do the following:

1. Create string arrays to store student registration number and name
2. Create double arrays to store Midterm marks and Final exam marks
3. Create two two-dimensional arrays for at least six quizzes and six assignments

4. Calculate the final score for each student by adding Midterm, Quizzes Total, Assignments Total and Final exam score. Weightages are: Midterm 25%, Quizzes 10%, Assignments 15%, and Final Exam 50%.
5. Assign Grade and GPA to each student based on the final score, for reference use Table 3.
6. Your program should also ask the user if he/she wants to select best of four quizzes and assignments and print alternate grades.

Note: Input validation is mandatory, negative values are not allowed.

Table 3: Grading criteria

Grade	Points	100
A+	4.00	>= 90
A	4.00	>= 86
A-	3.67	>= 82
B+	3.33	>= 78
B	3.00	>= 74
B-	2.67	>= 70
C+	2.33	>= 66
C	2.00	>= 62
C-	1.67	>= 58
D+	1.33	>= 54
D	1.00	>= 50
F	0.00	< 50

Problem 5: Snake game **[50 Marks]**

Simulate a two player snake board game as shown in Figure 1 below.

Figure 1: Snake board game

You are required to do the following:

1. Create a snake board of size M rows and N columns. As C++ is a row major language, game will start at [M-1][0] and end at [0][N-1] (*for odd M*) or at [0][0] (*for even M*).
2. Randomly generate N-1 snakes on the board. In order to generate snake you only need to know head and tail of the snake. Make sure that both head and tail are on the board. Moreover, if head is on row (M_i) and tail is on row (M_j) then *i* will always be less than *j*.
3. Similarly, generate N-1 ladders on the board.

4. In order to start both the player need a six on the dice. Once the game is started display the output on the dice and wait for key press (*you can use getch()*) before second player's turn.
5. Game will go on until one player wins the game.
6. In case player lands on a snake's head it will come down to its tail, here you need to display a message "oops, snake got you!!!"
7. In case player lands on the bottom of the ladder it will climb the ladder, here you need to display a message "you got lucky"

Note: Input validation is mandatory when reading taking input size of the board. Lands on snake's head / lands on bottom of the ladder: The term "**lands on**" means the final position of each player after each turn. So crossing a snake's head or bottom of ladder does not mean that you have landed.