National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures Lab	Course Code:	CL 2001
Program:	BS(CS)	Semester:	Fall 2021
Duration:	3 hours	Total Marks:	60
Paper Date:	November 15, 2021	Weight	-
Section:	BCS 3E	Page(s):	3
Exam:	Midterm	Roll No:	

Instruction/Notes:

- **Submission details: Paste your answers under the respective part of question in this Final Exam file. You also need to submit the .h and .cpp files. Take photograph(s) of dry run and paste it in the word document under question statement where needed. Archive all your files and name this file as your roll number and please upload this file on classrooms.**

Problem 1 [20 marks]

A super store is currently managing all the records manually and they want to automate their system. Your task is to provide them a solution to this problem. Each entry should contain **invoice number, date, and the total bill**. Create a program which handles the basic operations such as **insert record, update record, delete record, and show record.** Insert and Update functions should receive invoice number, date, and bill from main. The only difference between insert and main is that Insert is going to insert each record whereas Update function should only update the record if the provided invoice number already exist otherwise display a message "**Invoice Number does not exist**". Similarly delete function should receive the invoice number and if the invoice number exist than delete the record otherwise display a message "Invoice Number does not exist". Remember there are huge number of entries and user frequently wants to see the record of last 3 entries so **choose the most suitable data structure** in which you can traverse in both forward and backward. To display last 3 entries to the user, create a separate function **show Recent Data**. This function should display last 3 entries on console.

Inside main: Insert 6 entries one after the other in the same sequence,
Insert (2, "10-11-2021", 18000)
Insert (3, "10-11-2021", 1600)
Insert (4, "11-11-2021", 19560)
Insert (6, "14-11-2021", 23780)
Insert (7, "15-11-2021", 54980)
Insert (8, "15-11-2021", 42970)
Insert the 7th entry **at head** with the following data
(1, "09-11-2021", 7560)
now show last 3 entries on the console by calling **show Recent Data**
call **show Data** function to display all the data console
Insert After (4, 5, "12-11-2021", 14650) //insert the record after invoice number 4.
Update (9, "13-11-2021", 54980)
Update (4, "18-11-2021", 67500)
Delete (6)
now show last 3 entries on the console by calling **show Recent Data**.
call **delete Record** which deletes all the records.

Problem 2 [20 marks]

We know that stack is used to manage function calls. Whenever a function is called, the local variables, and return address is pushed on to the stack. Once the function completes its execution the return address and local variables are popped from the stack. Your task is to provide an array-based stack implementation to manage the function calls.

Data Members and Methods:

string * arr (This array should be used to store the return address against each function call)
int size (size of array).
int count (count of elements pushed on stack).
parameterized constructor, it should receive the size and creates an array of that fixed size.
isFull function, it should return 1 if the stack is full otherwise 0.
push function, receive and inserts the element into the stack. If the stack is full call the reSize function.
isEmpty function, it should return 1 if the stack is empty otherwise 0.
pop function, removes the element from stack. if the stack is empty than return "-1".
reSize function which double the size of array.

Inside main:

create a stack of size 4 and push the following return addresses on stack.
Push ("1180AF")
Push ("1230CE")
Push ("1044AD")
Push ("9020BD")
Pop () //remove the element from stack and display the return address on console
Push ("1166")
Push ("1177")
getCount() //this function should return the count of elements in the stack
getSize() //this function should return the size of array.
remove all the elements from the stack by calling pop function and display the return address on console.

Problem 3 [20 marks]

Overload the two functions named as **check Palindrome**. The first one should have a **string** as parameter and the second one must have an **integer** value. This function should return true if the provided input is palindrome otherwise return false. Display the message on console whether the provided input is **"palindrome" or not.**

- Write a **main function** to test the following cases:
- checkPalindrome("1122")
- checkPalindrome("1212")
- checkPalindrome("1221")
- checkPalindrome("121")
- checkPalindrome("11")
- checkPalindrome("1")
- checkPalindrome(" ")
- checkPalindrome(1122)
- checkPalindrome(1212)
- checkPalindrome(1221)
- checkPalindrome(121)
- checkPalindrome(11)
- checkPalindrome(1)