Data Structures LAB 02 – BITF24M

In drive D: or some other suitable drive create a folder dsalab02 and change folder to that so that the prompt should be like **D:\rollno\dsalab02>.**

Make sure that command g++ is recognized at the computer you have, also check the Notepad++.

To create or edit abc.cpp, use command: **D:\rollno\dsalab02>**`start notepad++ abc.cpp`
To compile abc.cpp, use command: **D:\rollno\dsalab02>**`g++ abc.cpp -o abc.exe`
To run binary of abc.cpp, the abc.exe, use command: **D:\rollno\dsalab02>**`abc.exe`

You may use any available IDE to complete the tasks.

Submission Protocol

All C++ (.cpp) files must be compressed into a ZIP file (not RAR or any other format) named with your roll number followed by Lab 02 (e.g., BITF24M598-Lab02.zip). Additionally, submit a document containing the code for the first 14 tasks. The document should be named similarly to the ZIP file and must include only your code arranged task-wise. Both the ZIP file and the document must be attached to an email and sent to all the specified email addresses no later than Saturday, 22 February 2026, at 11:59 PM.

bcsf23m524@pucit.edu.pk
bsdsf23a008@pucit.edu.pk
bsdsf23a024@pucit.edu.pk
bsdsf23m003@pucit.edu.pk
bsdsf23m005@pucit.edu.pk
bsdsf24m017@pucit.edu.pk

Tasks set 1 (5 mark for each)

1. You are provided with code file scenario.cpp, understand it. I have made these tasks simpler and generally no boundary conditions checks/handling code are required. If they are required, note when solving following tasks.
2. Copy scenario.cpp as task02.cpp and as guided in main, write code to update all values with their square roots.
3. Copy scenario.cpp as task03.cpp and as guided in main, write code to add a value 55.55 at the end of the list.
4. Copy scenario.cpp as task04.cpp and as guided in main, write code to add a value 22.22 at the beginning of the list.

5. Copy scenario.cpp as task05.cpp and as guided in main, write code to add a value 44.44 just after existing value 20 in the list.
6. Copy scenario.cpp as task06.cpp and as guided in main, write code to add a value 77.77 just before the existing value 70 in the list.
7. Copy scenario.cpp as task07.cpp and as guided in main, write code to remove 90 from the list. You should make sure to deallocate the memory of removed node guarantying no memory leak.
8. Copy scenario.cpp as task08.cpp and as guided in main, write code to remove first value from the list. You should make sure to deallocate the memory of removed node guarantying no memory leak.
9. Copy scenario.cpp as task09.cpp and as guided in main, write code to remove last value from the list. You should make sure to deallocate the memory of removed node guarantying no memory leak.
10. Copy scenario.cpp as task10.cpp and as guided in main, write code output (print) the minimum and maximum values from the list.

Tasks set 2 (10 mark for each)

In these tasks you must take care of the boundary cases.

11. Copy scenario.cpp as task11.cpp and update the makelist function, so that it asks user to enter 10 values to store in the list. Also, make a new function, showlist with similar code in main and makelist functions to display the list. Lastly, update the code in file to demonstrate the functions mentioned in this task.
12. Copy finished task11.cpp as task12.cpp and create a function void append(float x) to add value of its parameter at the end of the list. Also, create another function void insert(float x, int i) to insert x at zero based ith location in the list (node it should not update the ith value but add a new node with value x at the ith location). Lastly, update the code in file to demonstrate the functions mentioned in this task.
13. Copy finished task12.cpp as task13.cpp and create a function void remove(float x) to remove value of its parameter from the list. Also, create functions void removenextof(float x), void removepeviousof(float x), and void removeith(int i). The purpose of the functions is reflected in their names. Lastly, update the code in file to demonstrate the functions mentioned in this task.

Tasks set 3 (10 mark for each)

Again, in these tasks you must take care of the boundary cases.

14. Copy finished task11.cpp as task14.cpp having functions makelist and showlist. Add the following recursive function in the code file and test whether it works as replacement of the

showlist function. Later, update its code to output the list in reverse order as stored in the linked list and demonstrate its working.

```
void showrec(Node *head)
{
    if (head != nullptr)
    {
        cout << head->info << ", ";
        showrec(head->next);
    }
    else
    {
        cout << '\b' << '\b' << '.' << endl;
    }
    return;
}
```

15. You should compile, run and then understand the code provided in list.cpp file. Concentrate on class(es) and comments in main function. Now arrange/manage all your above tasks code as C++ class. The code in list.cpp should be your baseline, but if you can change the code totally to make class(es) it is fine as long as main is working according to comments in it.

Thank you for your **patience**