CL-2001

Data Structures

Lab # 4

|  |
| --- |
| Objectives:  1. Singly Linked List 2. Doubly Linked List |

**Note: Carefully read the following instructions (*Each instruction contains a weightage*)**

1. There must be a block of comments at start of every question's code by students; the block should contain brief description about functionality of code.
2. Comment on every function about its functionality.
3. Use understandable name of variables.
4. Proper indentation of code is essential.
5. Make separate .cpp files for all tasks and use this format **22F-1234\_Task1.cpp.**
6. First think about statement problems and then write/draw your logic on copy.
7. After copy pencil work, code the problem statement on C++ compiler.
8. Make a Microsoft Word file and paste all of your C++ code with all possible screenshots of every **task output in MS word and submit .cpp file with word file**.
9. Please submit your file in this format **22F-1234\_L1**.
10. Do not submit your assignment **after the deadline**.
11. **Do not copy code from any source otherwise you will be penalized with negative marks.**

|  |
| --- |
| **Problem 1:** |

Write a C++ program to manage a singly linked list. Define a class for a linked list node with an integer data field and a pointer to the next node.

1. Implement a function **createNode** that dynamically allocates memory for a new node, initializes its data, and sets the next pointer to null.
2. Implement a function **insertAtBeginning** that takes an integer as input and inserts a new node with that value at the beginning of the linked list. If the list is initially empty, this node should be the only node in the list.
3. Implement a function **insertAtEnd** that inserts a new node with a given value at the end of the linked list. If the list is empty, this node should be the only node in the list.
4. Implement a function **deleteFirstNode** that deletes the first node from the list. If the list becomes empty after deletion, ensure the pointer to the head of the list is properly updated.
5. Implement a function **searchAndPrint** that takes an integer value as input, searches for it in the linked list, and prints its position (0-based) if found. If the value is not present, print "Value not found".
6. Finally, write a function **printList** that traverses the entire linked list and prints its elements.

You can assume that the linked list is initially empty. Provide a C++ program with the above functions and a sample usage demonstrating all the functionalities.

|  |
| --- |
| **Problem 2:** |

Consider a C++ program to manage a doubly linked list. Define a class for a doubly linked list node with an integer data field and pointers to the next and previous nodes.

1. Write a function **createNode** that dynamically allocates memory for a new node, initializes its data, and sets both next and previous pointers to null.
2. Implement a function **insertAtEnd** that inserts a new node with a given value at the end of the doubly linked list. If the list is empty, this node should be the only node in the list.
3. Create a function **printList** that traverses the entire doubly linked list both forwards and backwards and prints its elements.
4. Write a function **reverseList** that reverses the order of nodes in the doubly linked list. The head of the list should be properly updated after the reversal.

Provide a C++ program with the above functions and a sample usage demonstrating the creation of nodes and reversal of the list.

|  |
| --- |
| **Problem 3: (Optional)** |

Design a program using a doubly linked list to manage the student enrollment system. Each student should be represented as a node in the doubly linked list with the following attributes:

Student ID

Student Name

Courses Enrolled

You need to implement the following operations:

* **Enroll a Student:** Implement a function to add a new student to the enrollment system. The student's ID and name should be provided as input.
* **Register for Courses**: Implement a function to allow a student to register for courses. The student provides their ID and a list of course codes.
* **Unregister from Courses:** Implement a function to allow a student to unregister from courses. The student provides their ID and the course codes they wish to drop.
* **List All Students:** Create a function to list all enrolled students along with their IDs and the courses they are registered for.
* **Search for a Student:** Implement a function that allows users to search for a student by ID or name. If the student is found, display their details including the courses they are enrolled in.

Your program should use a doubly linked list to efficiently manage the enrolled students and their course registrations. It should provide an easy way to add, modify, and search for student information