Data Structure and Algorithms Lab

LAB PROGRAM 1

1. Write a program that allows the user to insert an element at the top of the stack, remove the element currently at the top, view the top element without removing it, and display all elements present in the stack.
    - Handle stack overflow and stack underflow conditions with appropriate error messages.
    - Accept the maximum size of the stack as user input.
    - Display the time complexity of each operation.
    - Execute the program for different stack sizes and record the execution times for push and pop operations to analyze performance.

2. Write a program that allows the user to add an element to the end of the queue, remove an element from the front, view the element at the front without removing it, and display all elements currently in the queue.
    - Handle queue overflow and underflow conditions with appropriate error messages.
    - Accept the maximum size of the queue as user input.
    - Display the time complexity of each operation.
    - Execute the program for different queue sizes and record the execution times for inserting and removing elements to analyze performance.

Additional Questions:

1. Write a program to simulate **undo** and **redo** operations in a text editor using two stacks. The program should allow the user to perform an action (such as typing a word or making a change), undo the most recent action by moving it from the “done” stack to the “undone” stack, and redo an undone action by moving it back to the “done” stack. It should handle cases where there are no actions to undo or redo, displaying appropriate messages. Finally, execute the program with a sequence of operations to demonstrate and verify its correctness, and analyze the time complexity of each operation.

2. Write a program to simulate a task scheduling system using a queue. The program should allow tasks to be added to the end of the queue in the order they arrive, process tasks from the front of the queue, and display the list of pending tasks at any time. It should handle cases where the queue is empty when attempting to process a task, as well as cases where the queue reaches its maximum capacity, by displaying appropriate messages. Finally, execute the program with a sequence of task additions and completions to demonstrate and verify its correctness, and analyze the time complexity of each operation.