Assignment6

StudentName: Keshav Singla **UID:22BCS13486**
Branch:CSE **Section/Group:22BCS_IOT_612**
Semester: 6th **DateofPerformance:17/03/25**
Subject Name: Advance prog. Lab **Subject Code: 22CSP-351**

Q1)ImplementQueueusingStack

- **Code:**

```
#include <stack>
using namespace std;

class QueueUsingStack {
    stack<int> s1, s2;
public:
    void enqueue(int x) {
        s1.push(x);
    }

    int dequeue() {
        if (s2.empty()) {
            while (!s1.empty()) {
                s2.push(s1.top());
                s1.pop();
            }
        }
        if (s2.empty()) {
            throw runtime_error("Queue is empty");
        }
        int top = s2.top();
        s2.pop();
        return top;
    }
};
```

Q2)Implement Stack using Queue

- **Code:**

```
#include <queue>
using namespace std;

class StackUsingQueue {
    queue<int> q1, q2;
public:
    void push(int x) {
        q2.push(x);
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }
        swap(q1, q2);
    }

    int pop() {
        if (q1.empty()) {
            throw runtime_error("Stack is empty");
        }
        int top = q1.front();
        q1.pop();
        return top;
    }

    int top() {
        if (q1.empty()) {
            throw runtime_error("Stack is empty");
        }
        return q1.front();
    }

    bool empty() {
        return q1.empty();
    }
};
```

Q3) <u>Deque using Queue</u>

- **Code:**

```
#include <deque>
```

```
using namespace std;

class DequeUsingQueue {
    deque<int> dq;
public:
    void pushFront(int x) {
        dq.push_front(x);
    }

    void pushBack(int x) {
        dq.push_back(x);
    }

    int popFront() {
        if (dq.empty()) {
            throw runtime_error("Deque is empty");
        }
        int front = dq.front();
        dq.pop_front();
        return front;
    }

    int popBack() {
        if (dq.empty()) {
            throw runtime_error("Deque is empty");
        }
        int back = dq.back();
        dq.pop_back();
        return back;
    }

    bool empty() {
        return dq.empty();
    }
};
```

Q4)ImplementStackusingLinkedList

- **Code:**

```
class Solution {
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(nullptr) {}
    };
    Node* top;

public:
    Solution() {
```

```
        top = nullptr;
    }

    void push(int x) {
        Node* newNode = new Node(x);
        newNode->next = top;
        top = newNode;
    }

    int pop() {
        if (!top) {
            throw runtime_error("Stack Underflow");
        }
        int data = top->data;
        Node* temp = top;
        top = top->next;
        delete temp;
        return data;
    }

    int peek() {
        if (!top) {
            throw runtime_error("Stack is empty");
        }
        return top->data;
    }

    bool isEmpty() {
        return top == nullptr;
    }
};
```

Q5)QueueImplementationusingLinkedList

- **Code:**

```
class Solution {
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(nullptr) {}
    };
    Node *front, *rear;

public:
    Solution() {
        front = rear = nullptr;
    }
```

```
    void enqueue(int x) {
        Node* newNode = new Node(x);
        if (!rear) {
            front = rear = newNode;
            return;
        }
        rear->next = newNode;
        rear = newNode;
    }

    int dequeue() {
        if (!front) {
            throw runtime_error("Queue Underflow");
        }
        int data = front->data;
        Node* temp = front;
        front = front->next;
        if (!front) rear = nullptr;
        delete temp;
        return data;
    }

    bool isEmpty() {
        return front == nullptr;
    }
};
```