Name: Sahil Mangla
Roll No: 1024030359

1.
```
#include <iostream>
using namespace std;
#define SIZE 100
class Queue {
    int arr[SIZE], front, rear;
public:
    Queue() { front = -1; rear = -1; }
    bool isEmpty() { return front == -1; }
    bool isFull() { return (rear + 1) % SIZE == front; }
   void enqueue(int x) {
        if (rear == SIZE - 1) {
            cout << "Queue is Full" << endl;
            return;
        }
        if (front == -1) front = 0;
        arr[++rear] = x;
        cout << x << " inserted" << endl;
    }
   void dequeue() {
    if (front == -1 || front > rear) {
        cout << "Queue is Empty" << endl;
        return;
    }
    cout << arr[front] << " removed" << endl;
    front++;
    }
    void display() {
        if (isEmpty()) {
            cout << "Queue is Empty" << endl;
            return;
        }
        cout << "Queue: ";
        int i = front;
        while (true) {
            cout << arr[i] << " ";
            if (i == rear) break;
            i = (i + 1) % SIZE;
        }
        cout << endl;
    }
    void peek() {
        if (isEmpty()) cout << "Queue is Empty" << endl;
        else cout << "Front element: " << arr[front] << endl;
    }
};
```

```
int main() {
    Queue q;
    int ch, val;
    do {
        cout << "\n1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n0. Exit\n";
        cout << "Enter choice: ";
        cin >> ch;
        switch (ch) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: cout << (q.isEmpty() ? "Queue is Empty" : "Queue is Not Empty") << endl; break;
            case 4: cout << (q.isFull() ? "Queue is Full" : "Queue is Not Full") << endl; break;
            case 5: q.display(); break;
            case 6: q.peek(); break;
            case 0: cout << "Exiting..." << endl; break;
            default: cout << "Invalid choice" << endl;
        }
    } while (ch != 0);
}
```

```
2.#include <bits/stdc++.h>
using namespace std;
#define SIZE 5
class CircularQueue
{
    int arr[SIZE];
    int front, rear;
public:
    CircularQueue()
    {
        front = -1;
        rear = -1;
    };
    bool isFull()
    {
        return (front == 0 && rear == SIZE - 1 || rear + 1 == front);
    }
    bool isEmpty()
    {
        return front == -1;
    }
    void enqueue(int x)
    {
        if (isFull())
            cout << "Queue is Full" << endl;
        rear = (rear + 1) % SIZE;
        if (front == -1)
            front = 0;
        arr[rear] = x;
        cout << x << " Inserted in the queue" << endl;
    }
    void dequeue()
    {
        if (isEmpty())
            cout << "Queue is Empty" << endl;
        if (front == rear)
        {
            front = -1;
            rear = -1;
        }
        else
        {
            cout << arr[front] << " dequeued" << endl;
            front = (front + 1) % SIZE;
        }
    }
    void peek()
    {
        if (isEmpty())
        {
            cout << "Queue is Empty\n";
        }
        else
        {
            cout << "Front element: " << arr[front] << "\n";
        }
    }
    void display()
    {
        if (isEmpty())
        {
            cout << "Queue is Empty\n";
            return;
        }
        cout << "Queue Elements: ";
        for (int i = front; i <= rear; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};
int main()
{
    CircularQueue q;
    int choice, value;
    do
    {   cout << "\n--- Circular Queue Menu ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit\n";
        cout << "Enter your choice: ";
        cin>>choice;
        switch (choice)
        {
```

```
        case 1:
            cout << "Enter value: ";
            cin >> value;
            q.enqueue(value);
            break;
        case 2:
            q.dequeue();
            break;
        case 3:
            q.peek();
            break;
        case 4:
            q.display();
            break;
        case 5:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice\n";
        }
    } while (choice != 0);
    return 0;
}
```

3.#include<bits/stdc++.h>

```
using namespace std;
int main(){
    queue<int> q;
    queue<int> q1;
    queue<int> q2;
    int arr[] = {4, 7, 11, 20, 5, 9};
    int n = 6;
    for (int i = 0; i < n; i++) {
        q.push(arr[i]);
    }
    for(int i=0;i<n/2;i++){
        q1.push(q.front());
        q.pop();
    }
    while(!q.empty()){
        q2.push(q.front());
        q.pop();
    }

    while (!q1.empty() && !q2.empty()) {
        q.push(q1.front());
        q1.pop();
        q.push(q2.front());
        q2.pop();
    }
    cout<<"Interleaved Elements are: ";
    while(!q.empty()){
        cout<<q.front()<<" ";
        q.pop();
    }cout<<endl;
    return 0;
}
```
4.

```
#include <iostream>
#include <queue>
#include <unordered_map>
using namespace std;
void firstNonRepeating(string str) {
    queue<char> q;
    unordered_map<char, int> freq;
    for (char ch : str) {
        q.push(ch);
        freq[ch]++;
        while (!q.empty() && freq[q.front()] > 1) {
            q.pop();
        }
        if (q.empty())
            cout << -1 << " ";
        else
            cout << q.front() << " ";
    }
    cout << endl;
}
int main() {
    string input;
    cout << "Enter string: ";
    getline(cin, input);
    string str = "";
    for (char ch : input) {
        if (ch != ' ') str += ch;
    }
    firstNonRepeating(str);
    return 0;
}
```
5 (a)

```
#include <iostream>
#include <queue>
using namespace std;
class Stack {
    queue<int> q1, q2;
public:
    void push(int x) {
        // Step 1: Enqueue to q2
        q2.push(x);
        // Step 2: Move all elements from q1 to q2
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }
        // Step 3: Swap q1 and q2
        swap(q1, q2);
    }
    void pop() {
        if (q1.empty()) {
            cout << "Stack is empty\n";
            return;
        }
        q1.pop();
    }
    int top() {
        if (q1.empty()) {
            cout << "Stack is empty\n";
            return -1;
        }
        return q1.front();
    }
    bool empty() {
        return q1.empty();
    }
};
int main() {
    Stack st;
    st.push(10);
    st.push(20);
    st.push(30);
    cout << "Top: " << st.top() << endl; // 30
    st.pop();
    cout << "Top after pop: " << st.top() << endl; // 20
    return 0;
}
5(b)#include <iostream>
#include <queue>
using namespace std;
class Stack {
    queue<int> q;
public:
    void push(int x) {
        int size = q.size();
        q.push(x);
        // Rotate: move all old elements behind new element
        for (int i = 0; i < size; i++) {
            q.push(q.front());
            q.pop();
        }
    }
    void pop() {
        if (q.empty()) {
            cout << "Stack is empty\n";
            return;
        }
        q.pop();
    }
    int top() {
        if (q.empty()) {
            cout << "Stack is empty\n";
            return -1;
        }
        return q.front();
    }
    bool empty() {
        return q.empty();
    }
};
int main() {
    Stack st;
    st.push(10);
    st.push(20);
    st.push(30);
    cout << "Top: " << st.top() << endl; // 30
    st.pop();
    cout << "Top after pop: " << st.top() << endl; // 20
    return 0;
}
```