

Assignment – 4

Ayush garg(1024030878)

Q1

```
1 #include <iostream>
2 using namespace std;
3 #define MAX 100
4 class Queue {
5     int arr[MAX];
6     int front, rear;
7 public:
8     Queue() { front = rear = -1; }
9
10    bool isEmpty() {
11        return front == -1;
12    }
13    bool isFull() {
14        return rear == MAX - 1;
15    }
16    void enqueue(int x) {
17        if (isFull()) { cout << "Queue Overflow\n"; return; }
18        if (front == -1) front = 0;
19        arr[++rear] = x;
20        cout << x << " enqueued\n";
21    }
22    void dequeue() {
23        if (isEmpty()) { cout << "Queue Underflow\n"; return; }
24        cout << arr[front] << " dequeued\n";
25        if (front == rear) front = rear = -1;
26        else front++;
27    }
28    void peek() {
29        if (isEmpty()) { cout << "Queue is Empty\n"; return; }
30        cout << "Front element: " << arr[front] << "\n";
31    }
32    void display() {
33        if (isEmpty()) { cout << "Queue is Empty\n"; return; }
34        cout << "Queue elements: ";
35        for (int i = front; i <= rear; i++) cout << arr[i] << " ";
36        cout << "\n";
37    }
38 };
39
40 int main() {
41     Queue q;
42     int choice, val;
43     while (true) {
44         cout << "\n--- SIMPLE QUEUE MENU ---\n";
45         cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";
46         cout << "Enter choice: "; cin >> choice;
47         switch (choice) {
48             case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
49             case 2: q.dequeue(); break;
50             case 3: cout << (q.isEmpty() ? "Queue is Empty\n" : "Queue not Empty\n"); break;
51             case 4: cout << (q.isFull() ? "Queue is Full\n" : "Queue not Full\n"); break;
52             case 5: q.display(); break;
53             case 6: q.peek(); break;
54             case 7: exit(0);
55             default: cout << "Invalid choice\n";
56         }
57     }
58 }
```

```
--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 23
23 enqueued

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 43
43 enqueued

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue elements: 23 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 3
Queue not Empty

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 4
Queue not Full

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 2
23 dequeued

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 5
Queue elements: 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
```

```

33     if (isEmpty()) { cout << "Queue is Empty\n"; return; }
34     cout << "Queue elements: ";
35     for (int i = front; i <= rear; i++) cout << arr[i] << " ";
36     cout << "\n";
37 }
38 };
39
40 int main() {
41     Queue q;
42     int choice, val;
43     while (true) {
44         cout << "\n--- SIMPLE QUEUE MENU ---\n";
45         cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";
46         cout << "Enter choice: "; cin >> choice;
47         switch (choice) {
48             case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
49             case 2: q.dequeue(); break;
50             case 3: cout << (q.isEmpty() ? "Queue is Empty\n" : "Queue not Empty\n"); break;
51             case 4: cout << (q.isFull() ? "Queue is Full\n" : "Queue not Full\n"); break;
52             case 5: q.display(); break;
53             case 6: q.peek(); break;
54             case 7: exit(0);
55             default: cout << "Invalid choice\n";
56         }
57     }
58 }

```

```

5. Display
6. Peek
7. Exit
Enter choice: 5
Queue elements: 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 6
Front element: 43

--- SIMPLE QUEUE MENU ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 7

```

=== Code Execution Successful ===

Q2

```

1 #include <iostream>
2 using namespace std;
3 #define MAX 5
4 class CircularQueue {
5     int arr[MAX];
6     int front, rear;
7 public:
8     CircularQueue() { front = rear = -1; }
9
10    bool isEmpty() { return front == -1; }
11    bool isFull()
12    { return (front == 0 && rear == MAX - 1) || (rear + 1 == front); }
13
14    void enqueue(int x) {
15        if (isFull()) { cout << "Queue Overflow\n"; return; }
16        if (isEmpty()) front = rear = 0;
17        else rear = (rear + 1) % MAX;
18        arr[rear] = x;
19        cout << x << " enqueued\n";
20    }
21
22    void dequeue() {
23        if (isEmpty()) { cout << "Queue Underflow\n"; return; }
24        cout << arr[front] << " dequeued\n";
25        if (front == rear) front = rear = -1;
26        else front = (front + 1) % MAX;
27    }
28
29    void peek() {
30        if (isEmpty()) { cout << "Queue Empty\n"; return; }
31        cout << "Front element: " << arr[front] << "\n";
32    }
33
34    void display() {
35        if (isEmpty()) { cout << "Queue Empty\n"; return; }
36        cout << "Queue: ";
37        int i = front;
38        while (true) {
39            cout << arr[i] << " ";
40            if (i == rear) break;
41            i = (i + 1) % MAX;
42        }
43        cout << "\n";
44    }
45
46    int main() {
47        CircularQueue q;
48        int choice, val;
49        while (true) {
50            cout << "\n--- Circular Queue Menu ---\n";
51            cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";
52            cout << "Enter choice: "; cin >> choice;
53            switch (choice) {
54                case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
55                case 2: q.dequeue(); break;
56                case 3: cout << (q.isEmpty() ? "Queue Empty\n" : "Queue not Empty\n"); break;

```

```

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 23
23 enqueued

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 1
Enter value: 67
67 enqueued

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 6
Front element: 23

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 2
23 dequeued

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
Enter choice: 3

```

```

37         if (i == iMax) break;
38         i = (i + 1) % MAX;
39     }
40     cout << "\n";
41 }
42 };
43
44 int main() {
45     CircularQueue q;
46     int choice, val;
47     while (true) {
48         cout << "\n--- Circular Queue Menu ---\n";
49         cout << "1. Enqueue\n2. Dequeue\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";
50         cout << "Enter choice: "; cin >> choice;
51         switch (choice) {
52             case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
53             case 2: q.dequeue(); break;
54             case 3: cout << (q.isEmpty() ? "Queue Empty\n" : "Queue not Empty\n"); break;
55             case 4: cout << (q.isFull() ? "Queue Full\n" : "Queue not Full\n"); break;
56             case 5: q.display(); break;
57             case 6: q.peek(); break;
58             case 7: exit(0);
59             default: cout << "Invalid choice\n";
60         }
61     }
62 }

```

Queue not Empty

--- Circular Queue Menu ---

1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit

Enter choice: 4

Queue not Full

--- Circular Queue Menu ---

1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit

Enter choice: 5

Queue: 67

--- Circular Queue Menu ---

1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit

Enter choice: 4

Queue not Full

--- Circular Queue Menu ---

1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit

Enter choice: 5

Queue: 67

--- Circular Queue Menu ---

1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit

Enter choice: 7

=== Code Execution Successful ===

Q3

```

1 #include <iostream>
2 #include <queue>
3 #include <stack>
4 using namespace std;
5
6 void interleaveQueue(queue<int>& q) {
7     if (q.size() % 2 != 0) { cout << "Queue has odd size, cannot interleave\n"; return; }
8
9     int half = q.size() / 2;
10    queue<int> firstHalf;
11
12    for (int i = 0; i < half; i++) {
13        firstHalf.push(q.front());
14        q.pop();
15    }
16
17    while (!firstHalf.empty()) {
18        q.push(firstHalf.front()); firstHalf.pop();
19        q.push(q.front()); q.pop();
20    }
21 }
22
23 int main() {
24     queue<int> q;
25     q.push(4); q.push(7); q.push(11); q.push(20); q.push(5); q.push(9);
26
27     interleaveQueue(q);
28
29     cout << "Output: ";
30     while (!q.empty()) { cout << q.front() << " "; q.pop(); }
31     cout << endl;
32 }

```

Output: 4 20 7 5 11 9

=== Code Execution Successful ===

Q4

```

1 #include <iostream>
2 #include <queue>
3 #include <unordered_map>
4 using namespace std;
5
6 void firstNonRepeating(string str) {
7     queue<char> q;
8     unordered_map<char, int> freq;
9
10    for (char c : str) {
11        freq[c]++;
12        q.push(c);
13
14        while (!q.empty() && freq[q.front()] > 1) q.pop();
15
16        if (q.empty()) cout << "-1 ";
17        else cout << q.front() << " ";
18    }
19    cout << endl;
20 }
21
22 int main() {
23     string str = "aabc";
24     cout << "Input: " << str << endl;
25     cout << "Output: ";
26     firstNonRepeating(str);
27 }

```

Input: aabc
Output: a -1 b b

=== Code Execution Successful ===

Q5(a)

```

1 #include <iostream>
2 #include <queue>
3 using namespace std;
4
5 class Stack {
6     queue<int> q1, q2;
7 public:
8     void push(int x) {
9         q2.push(x);
10        while (!q1.empty()) {
11            q2.push(q1.front()); q1.pop();
12        }
13        swap(q1, q2);
14    }
15
16    void pop() {
17        if (q1.empty()) { cout << "Stack Empty\n"; return; }
18        cout << q1.front() << " popped\n";
19        q1.pop();
20    }
21
22    void top() {
23        if (q1.empty()) { cout << "Stack Empty\n"; return; }
24        cout << "Top: " << q1.front() << "\n";
25    }
26 };
27
28 int main() {
29     Stack s;
30     s.push(10);
31     s.push(20);
32     s.push(30);
33     s.top();
34     s.pop();
35     s.top();
36 }

```

Top: 30
30 popped
Top: 20

=== Code Execution Successful ===

Q5(b)

```

1 #include <iostream>
2 #include <queue>
3 using namespace std;
4
5 class Stack {
6     queue<int> q;
7 public:
8     void push(int x) {
9         int n = q.size();
10        q.push(x);
11        for (int i = 0; i < n; i++) {
12            q.push(q.front());
13            q.pop();
14        }
15    }
16
17    void pop() {
18        if (q.empty()) { cout << "Stack Empty\n"; return; }
19        cout << q.front() << " popped\n";
20        q.pop();
21    }
22
23    void top() {
24        if (q.empty()) { cout << "Stack Empty\n"; return; }
25        cout << "Top: " << q.front() << "\n";
26    }
27 };
28
29 int main() {
30     Stack s;
31     s.push(10);
32     s.push(20);
33     s.push(30);
34     s.top();
35     s.pop();
36     s.top();
37 }

```

Top: 30
30 popped
Top: 20

=== Code Execution Successful ===

