

- 1) Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

```
Solution:- #include <iostream>

using namespace std;

class Queue {
    int front, rear, size;
    int *arr;
public:
    Queue(int n) {
        size = n;
        arr = new int[size];
        front = rear = -1;
    }

    bool isFull() {
        return rear == size - 1;
    }

    bool isEmpty() {
        return front == -1 || front > rear;
    }

    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue is FULL\n";
            return;
        }
        if (front == -1) front = 0;
        arr[++rear] = x;
        cout << "Inserted: " << x << endl;
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue is EMPTY\n";
            return;
        }
        cout << "Deleted: " << arr[front++] << endl;
    }
}
```

```

void peek() {
    if (isEmpty()) cout << "Queue Empty\n";
    else cout << "Front element: " << arr[front] << endl;
}

void display() {
    if (isEmpty()) {
        cout << "Queue Empty\n";
        return;
    }
    cout << "Queue: ";
    for (int i = front; i <= rear; i++)
        cout << arr[i] << " ";
    cout << endl;
}
};

int main() {
    Queue q(10);
    int ch, val;

    while (true) {
        cout <<
"\n1.Enqueue  2.Dequeue  3.isEmpty  4.isFull  5.Display  6.Peek  7.Exit\n";
        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() ? "Empty\n" : "Not Empty\n"); break;
            case 4: cout << (q.isFull() ? "Full\n" : "Not Full\n"); break;
            case 5: q.display(); break;
            case 6: q.peek(); break;
            case 7: return 0;
        }
    }
}

```

**Output :-**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\DELL\OneDrive\Desktop\ds_lab> cd 'C:\Users\DELL\OneDrive\Desktop\ds_lab\ass_4\output'
PS C:\Users\DELL\OneDrive\Desktop\ds_lab\ass_4\output> & .\ques1.exe

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

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

2) Develop a menu driven program demonstrating the following operations on Circular Queues:

enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

**Solution:-**

```
#include <iostream>

using namespace std;

class CircularQueue {
    int front, rear, size;
    int *arr;
public:
    CircularQueue(int n) {
        size = n;
        arr = new int[n];
        front = rear = -1;
    }

    bool isEmpty() {
        return front == -1;
    }

    bool isFull() {
        return (rear + 1) % size == front;
    }

    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue FULL\n";
            return;
        }
    }
}
```

```

        if (front == -1) front = 0;
        rear = (rear + 1) % size;
        arr[rear] = x;
        cout << "Inserted " << x << endl;
    }

void dequeue() {
    if (isEmpty()) {
        cout << "Queue EMPTY\n";
        return;
    }
    cout << "Deleted " << arr[front] << endl;

    if (front == rear) front = rear = -1;
    else front = (front + 1) % size;
}

void peek() {
    if (isEmpty()) cout << "Empty\n";
    else cout << "Front: " << arr[front] << endl;
}

void display() {
    if (isEmpty()) {
        cout << "Empty\n"; return;
    }
    cout << "Queue: ";
    int i = front;
    while (true) {
        cout << arr[i] << " ";
        if (i == rear) break;
        i = (i + 1) % size;
    }
    cout << endl;
}

int main() {
    CircularQueue cq(10);
    int ch, val;

    while (true) {
        cout <<
"\n1.Enqueue  2.Dequeue  3.isEmpty  4.isFull  5.Display  6.Peek  7.Exit\n";
        cin >> ch;

```

```

        switch(ch) {
            case 1: cout << "Enter value: "; cin >> val; cq.enqueue(val); break;
            case 2: cq.dequeue(); break;
            case 3: cout << (cq.isEmpty() ? "Empty\n" : "Not Empty\n"); break;
            case 4: cout << (cq.isFull() ? "Full\n" : "Not Full\n"); break;
            case 5: cq.display(); break;
            case 6: cq.peek(); break;
            case 7: return 0;
        }
    }
}

```

#### Output:-

```

PS C:\Users\DELL\OneDrive\Desktop\ds lab> cd 'c:\Users\DELL\OneDrive\Desktop\ds lab\ass 4\output'
PS C:\Users\DELL\OneDrive\Desktop\ds lab\ass 4\output> & .\ques2.exe

1.Enqueue 2.Dequeue 3.isEmpty 4.isFull 5.Display 6.Peek 7.Exit
4
Not Full

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


```

#### 3) Write a program interleave the first half of the queue with second half.

Sample I/P: 4 7 11 20 5 9 Sample O/P: 4 20 7 5 11 9

**Solution:-**

```

#include <iostream>

#include <queue>
using namespace std;

void interleave(queue<int>& q) {
    int n = q.size();
    if (n % 2 != 0) {
        cout << "Queue must have even number of elements!\n";
        return;
    }

    queue<int> firstHalf;

```

```

        for (int i = 0; i < n/2; i++) {
            firstHalf.push(q.front());
            q.pop();
        }

        while (!firstHalf.empty()) {
            q.push(firstHalf.front());
            firstHalf.pop();

            q.push(q.front());
            q.pop();
        }
    }

int main() {
    queue<int> q;
    q.push(4); q.push(7); q.push(11); q.push(20); q.push(5); q.push(9);

    interleave(q);

    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
}

```

#### Output:-

```

PS C:\Users\OneDrive\Desktop\ds_lab> cd 'c:\Users\OneDrive\Desktop\ds_lab\ass_4\output'
PS C:\Users\OneDrive\Desktop\ds_lab\ass_4\output> & .\ques3.exe
4 20 7 5 11 9
PS C:\Users\OneDrive\Desktop\ds_lab\ass_4\output>

```

**4) Write a program to find first non-repeating character in a string using Queue.**

**Sample I/P:** a a b c **Sample O/P:** a -1 b b

```

Solution:- #include <iostream>
#include <queue>
using namespace std;

int main() {
    string s;
    cin >> s;

    queue<char> q;
    int freq[26] = {0};

    for (char c : s) {
        freq[c - 'a']++;
        q.push(c);

        while (!q.empty() && freq[q.front() - 'a'] > 1)
            q.pop();

        if (q.empty()) cout << "-1 ";
        else cout << q.front() << " ";
    }
}

```

**5) Write a program to implement a stack using (a) Two queues and (b) One Queue.**

**Solution:- Part-A**

```

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

class StackTwoQueues {
    queue<int> q1, q2;

public:
    void push(int x) {
        q2.push(x);
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }
    }

    int pop() {
        if (q1.empty())
            return -1;
        else {
            int val = q1.front();
            q1.pop();
            return val;
        }
    }

    int top() {
        if (q1.empty())
            return -1;
        else
            return q1.front();
    }

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

```

```

        q1.pop();
    }
    swap(q1, q2);
}

void pop() {
    if (q1.empty()) {
        cout << "Stack Empty\n"; return;
    }
    cout << "Popped: " << q1.front() << endl;
    q1.pop();
}

void top() {
    if (q1.empty()) cout << "Empty\n";
    else cout << "Top: " << q1.front() << endl;
}
};

```

```

B] #include <iostream>

#include <queue>
using namespace std;

class StackOneQueue {
    queue<int> q;

public:
    void push(int x) {
        q.push(x);
        int size = q.size();
        while (size > 1) {
            q.push(q.front());
            q.pop();
            size--;
        }
    }

    void pop() {
        if (q.empty()) {
            cout << "Stack Empty\n"; return;
        }
    }
};

```

```
    cout << "Popped: " << q.front() << endl;
    q.pop();
}

void top() {
    if (q.empty()) cout << "Empty\n";
    else cout << "Top: " << q.front() << endl;
}
};
```