

DSA Assignment 4

Q1.

```
#include <iostream>
using namespace std;
#define SIZE 5

class SimpleQueue {
    int arr[SIZE], front, rear;
public:
    SimpleQueue() { front = -1; rear = -1; }

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

    void enqueue(int x) {
        if (isFull()) cout << "Queue Full\n";
        else {
            if (front == -1) front = 0;
            arr[++rear] = x;
        }
    }

    void dequeue() {
        if (isEmpty()) cout << "Queue Empty\n";
        else {
            cout << "Deleted: " << arr[front] << endl;
            if (front == rear) front = rear = -1;
            else front++;
        }
    }

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

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

int main() {
    SimpleQueue q;
    int ch, val;
    while (1) {
        cout << "1-Enqueue\n2-Dequeue \n3-Display \n4-Peek \n5-Exit\n";
        cin >> ch;
        if (ch == 1) { cin >> val; q.enqueue(val); }
        else if (ch == 2) q.dequeue();
        else if (ch == 3) q.display();
        else if (ch == 4) q.peek();
        else break;
    }
}
```

```

    }
}

```

OUTPUT

1-Enqueue	1-Enqueue
2-Dequeue	2-Dequeue
3-Display	3-Display
4-Peek	4-Peek
5-Exit	5-Exit
1	4
12	Front: 2
1-Enqueue	1-Enqueue
2-Dequeue	2-Dequeue
3-Display	3-Display
4-Peek	4-Peek
5-Exit	5-Exit
2	5
Deleted: 12	
1-Enqueue	
2-Dequeue	
3-Display	
4-Peek	
5-Exit	
1	
2	
1-Enqueue	
2-Dequeue	
3-Display	
4-Peek	
5-Exit	
1	
3	
1-Enqueue	
2-Dequeue	
3-Display	
4-Peek	
5-Exit	
3	
2 2 3	

Q2.

```

#include <iostream>
using namespace std;
#define SIZE 5

class CircularQueue {
    int arr[SIZE], front, rear;
public:
    CircularQueue() { front = -1; rear = -1; }

    bool isEmpty() { return front == -1; }
    bool isFull() { return (rear + 1) % SIZE == front; }

    void enqueue(int x) {
        if (isFull()) cout << "Queue Full\n";
        else {
            if (front == -1) front = 0;
            rear = (rear + 1) % SIZE;

```

```

        arr[rear] = x;
    }
}

void dequeue() {
    if (isEmpty()) cout << "Queue Empty\n";
    else {
        cout << "Deleted: " << arr[front] << endl;
        if (front == rear) front = rear = -1;
        else front = (front + 1) % SIZE;
    }
}

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

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

};

int main() {
    CircularQueue q;
    int ch, val;
    while (1) {
        cout << "1-Enqueue 2-Dequeue 3-Display 4-Peek 5-Exit\n";
        cin >> ch;
        if (ch == 1) { cin >> val; q.enqueue(val); }
        else if (ch == 2) q.dequeue();
        else if (ch == 3) q.display();
        else if (ch == 4) q.peek();
        else break;
    }
}

```

OUTPUT

Same as q1

Q3.

```

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

void interleave(queue<int>& q) {
    int n = q.size();
    int half = n / 2;

```

```

queue<int> firstHalf;

for (int i = 0; i < half; 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

```

4 20 7 5 11 9 %

```

Q4.

```

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

void firstNonRepeating(string s) {
    unordered_map<char,int> freq;
    queue<char> q;

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

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

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

int main() {
    string s = "aabc";
    firstNonRepeating(s);
}

```

OUTPUT

```
a -1 b b %  
Q5(a).  
#include <iostream>  
#include <queue>  
using namespace std;  
  
class Stack {  
    queue<int> q1, q2;  
public:  
    void push(int x) {  
        q2.push(x);  
        while (!q1.empty()) {  
            q2.push(q1.front());  
            q1.pop();  
        }  
        swap(q1, q2);  
    }  
  
    void pop() {  
        if (!q1.empty()) q1.pop();  
        else cout << "Stack Empty\n";  
    }  
  
    int top() {  
        if (!q1.empty()) return q1.front();  
        return -1;  
    }  
  
    bool empty() { return q1.empty(); }  
};  
  
int main() {  
    Stack s;  
    s.push(10); s.push(20); s.push(30);  
    cout << s.top() << endl; // 30  
    s.pop();  
    cout << s.top() << endl; // 20  
}
```

OUTPUT

```
30  
20
```

Q5(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);  
        for (int i = 0; i < size; i++) {
```

```

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

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

int top() {
    if (!q.empty()) return q.front();
    return -1;
}

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

int main() {
    Stack s;
    s.push(10); s.push(20); s.push(30);
    cout << s.top() << endl; // 30
    s.pop();
    cout << s.top() << endl; // 20
}

```

OUTPUT

```

30
20

```