# Lab Assignment 4: Queues - Solutions

## Q1. Simple Queue

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

class Queue{
    int arr[10];
    int front, rear, capacity;
public:
    Queue(int c=10){front=-1; rear=-1; capacity=c;}
    int isEmpty(){return front==-1;}
    int isFull(){return (rear+1)%capacity==front;}
    void enqueue(int x){
        if(isFull()) return;
        if(front==-1) front=0;
        rear=(rear+1)%capacity;
        arr[rear]=x;
    }
    void dequeue(){
        if(isEmpty()) return;
        if(front==rear){front=-1; rear=-1;}
        else front=(front+1)%capacity;
    }
    void display(){
        if(isEmpty()) return;
        int i=front;
        while(1){
            cout<<arr[i]<<" ";
            if(i==rear) break;
            i=(i+1)%capacity;
        }
        cout<<endl;
    }
    void peek(){
        if(!isEmpty()) cout<<arr[front]<<endl;
    }
};

int main(){
    Queue q;
    q.enqueue(10); q.enqueue(20); q.enqueue(30);
    q.display(); q.peek();
    q.dequeue(); q.display();
}
```

## Q2. Circular Queue

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

class CQueue{
    int arr[10];
    int front, rear, capacity;
public:
    CQueue(int c=10){front=-1; rear=-1; capacity=c;}
    int isEmpty(){return front==-1;}
    int isFull(){return (rear+1)%capacity==front;}
    void enqueue(int x){
        if(isFull()) return;
        if(front==-1) front=0;
        rear=(rear+1)%capacity;
        arr[rear]=x;
    }
    void dequeue(){
        if(isEmpty()) return;
        if(front==rear){front=-1; rear=-1;}
        else front=(front+1)%capacity;
    }
    void display(){
        if(isEmpty()) return;
        int i=front;
        while(1){
            cout<<arr[i]<<" ";
            if(i==rear) break;
            i=(i+1)%capacity;
        }
        cout<<endl;
    }
    void peek(){
        if(!isEmpty()) cout<<arr[front]<<endl;
    }
};

int main(){
    CQueue cq;
    cq.enqueue(1); cq.enqueue(2); cq.enqueue(3);
    cq.display(); cq.peek();
    cq.dequeue(); cq.display();
}
```

## Q3. Interleave Queue

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

class Queue{
    int arr[50];
    int front, rear, capacity;
public:
    Queue(int c=50){front=-1; rear=-1; capacity=c;}
    int isEmpty(){return front==-1;}
    void enqueue(int x){
        if(rear==capacity-1) return;
        if(front==-1) front=0;
        arr[++rear]=x;
    }
    int dequeue(){
        if(isEmpty()) return -1;
        int x=arr[front];
        if(front==rear){front=-1; rear=-1;}
        else front++;
        return x;
    }
    int size(){return isEmpty()?0:rear-front+1;}
};

int main(){
    Queue q1,q2,q;
    int n,x; cin>>n;
    for(int i=0;i<n;i++){cin>>x; q1.enqueue(x);}
    for(int i=0;i<n/2;i++) q2.enqueue(q1.dequeue());
    while(!q1.isEmpty() && !q2.isEmpty()){
        q.enqueue(q2.dequeue());
        q.enqueue(q1.dequeue());
    }
    while(!q.isEmpty()) cout<<q.dequeue()<<" ";
}
```

## Q4. First Non-Repeating Character

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

class Queue{
    char arr[256];
    int f,r,capacity;
public:
    Queue(int c=256){f=-1;r=-1;capacity=c;}
    int isEmpty(){return f==-1;}
    void enqueue(char c){
        if(r==capacity-1) return;
        if(f==-1) f=0;
        arr[++r]=c;
    }
    char dequeue(){
        if(isEmpty()) return '\0';
        char c=arr[f];
        if(f==r){f=-1;r=-1;}
        else f++;
        return c;
    }
    char front(){return arr[f];}
};

int main(){
    string s; cin>>s;
    int freq[256]={0};
    Queue q;
    for(int i=0;i<s.size();i++){
        char c=s[i];
        freq[c]++; q.enqueue(c);
        while(!q.isEmpty() && freq[q.front()]>1) q.dequeue();
        if(q.isEmpty()) cout<<"-1 ";
        else cout<<q.front()<<" ";
    }
}
```

## Q5a. Stack Using Two Queues

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

class Queue{
    int arr[50];
    int f,r,capacity;
public:
    Queue(int c=50){f=-1;r=-1;capacity=c;}
    int isEmpty(){return f==-1;}
    void enqueue(int x){
        if(r==capacity-1) return;
        if(f==-1) f=0;
        arr[++r]=x;
    }
    int dequeue(){
        if(isEmpty()) return -1;
        int x=arr[f];
        if(f==r){f=-1;r=-1;}
        else f++;
        return x;
    }
    int size(){return isEmpty()?0:r-f+1;}
};

class Stack{
    Queue q1,q2;
public:
    void push(int x){
        q2.enqueue(x);
        while(!q1.isEmpty()) q2.enqueue(q1.dequeue());
        Queue temp=q1; q1=q2; q2=temp;
    }
    void pop(){q1.dequeue();}
    int top(){return q1.isEmpty()?-1:q1.dequeue();}
};

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

## Q5b. Stack Using One Queue

```
#include<iostream>
```

```
using namespace std;

class Queue{
    int arr[50];
    int f,r,capacity;
public:
    Queue(int c=50){f=-1;r=-1;capacity=c;}
    int isEmpty(){return f==-1;}
    void enqueue(int x){
        if(r==capacity-1) return;
        if(f==-1) f=0;
        arr[++r]=x;
    }
    int dequeue(){
        if(isEmpty()) return -1;
        int x=arr[f];
        if(f==r){f=-1;r=-1;}
        else f++;
        return x;
    }
    int size(){return isEmpty()?0:r-f+1;}
};

class Stack{
    Queue q;
public:
    void push(int x){
        q.enqueue(x);
        for(int i=0;i<q.size()-1;i++) q.enqueue(q.dequeue());
    }
    void pop(){q.dequeue();}
    int top(){return q.isEmpty()?-1:q.dequeue();}
};

int main(){
    Stack s; s.push(5); s.push(15);
    cout<<s.top()<<endl;
    s.pop(); cout<<s.top()<<endl;
}
```