Data Structures and Algorithms

Lab Assignment 6

SUBMITTED BY:

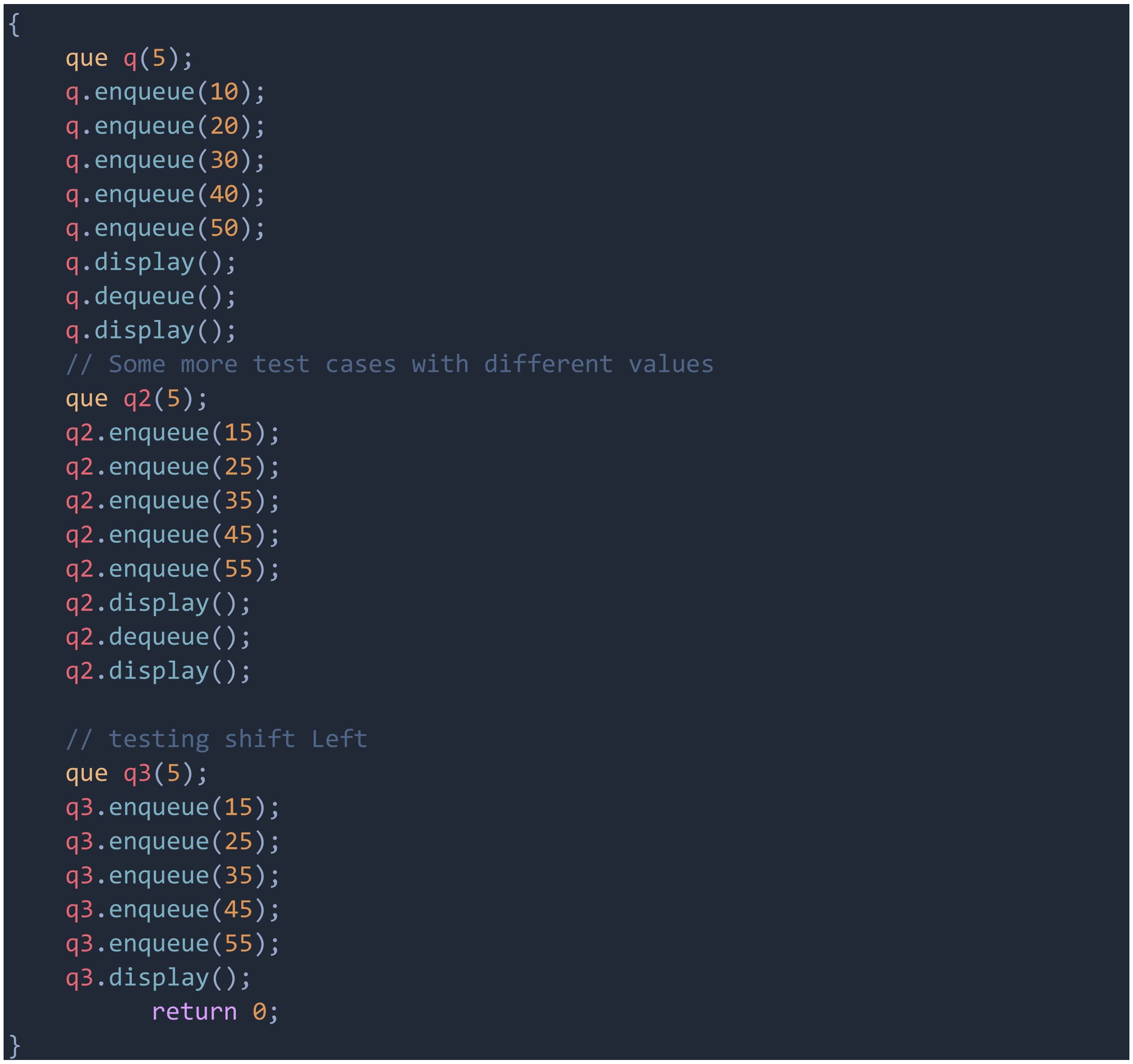
Hasaan Ahmad SP22-BSE-017

SUBMITTED TO: Sir Syed Ahmad Qasim

Static Queue And Deque Implementation

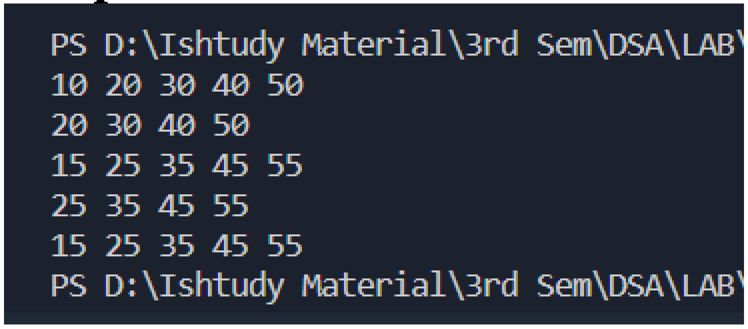
```cpp
#include <iostream>
using namespace std;

class que
{
public:
    int size;
    int *q;
    int front;
    int rear;
    que()
    {
        size = 10;
        q = new int[size];
        front = rear = -1;
    }
    que(int size)
    {
        this->size = size;
        q = new int[this->size];
        front = rear = -1;
    }
    bool isEmpty()
    {
        if (front == rear)
        {
            return true;
        }
        return false;
    }
    bool isFull()
    {
        if (rear == size - 1)
        {
            return true;
        }
        return false;
    }
    void enqueue(int x)
    {
        if (isFull())
        {
            cout << "Queue is Full" << endl;
        }
        else
        {
            rear++;
            q[rear] = x;
        }
    }
    int dequeue()
    {
        int x = -1;
        if (isEmpty())
        {
            cout << "Queue is Empty" << endl;
        }
        else
        {
            front++;
            x = q[front];
        }
        return x;
    }
    void display()
    {
        for (int i = front + 1; i <= rear; i++)
        {
            cout << q[i] << " ";
        }
        cout << endl;
    }

    //  A method which also shifts the element left
    int deQue(){
        int x = -1;
        if(isEmpty()){
            cout << "Queue is Empty" << endl;
        }
        else{
            x = q[front + 1];
            for(int i = front + 1; i < rear; i++){
                q[i] = q[i + 1];
            }
            rear--;
        }
        return x;
    }
```

```cpp
    void shiftLeft(int front,int rear){
        for(int i = front + 1; i < rear; i++){
            q[i] = q[i + 1];
        }
        rear--;
    }

};

class Deque{
    public:
        int size;
        int *q;
        int front;
        int rear;
        Deque(){
            size = 10;
            q = new int[size];
            front = rear = -1;
        }
        Deque(int size){
            this->size = size;
            q = new int[this->size];
            front = rear = -1;
        }
        bool isEmpty(){
            if(front == rear){
                return true;
            }
            return false;
        }
        bool isFull(){
            if(rear == size - 1){
                return true;
            }
            return false;
        }
        void enqueueFront(int x){
            if(isFull()){
                cout << "Queue is Full" << endl;
            }
            else{
                for(int i = rear; i > front; i--){
                    q[i + 1] = q[i];
                }
                front++;
                q[front] = x;
                rear++;
            }
        }
        void enqueueRear(int x){
            if(isFull()){
                cout << "Queue is Full" << endl;
            }
            else{
                rear++;
                q[rear] = x;
            }
        }
        int dequeueFront(){
            int x = -1;
            if(isEmpty()){
                cout << "Queue is Empty" << endl;
            }
            else{
                x = q[front + 1];
                front++;
            }
            return x;
        }
        int dequeueRear(){
            int x = -1;
            if(isEmpty()){
                cout << "Queue is Empty" << endl;
            }
            else{
                x = q[rear];
                rear--;
            }
            return x;
        }
        void display(){
            for(int i = front + 1; i <= rear; i++){
                cout << q[i] << " ";
            }
            cout << endl;
        }
};

int main()
```



```
{
    que q(5);
    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);
    q.enqueue(40);
    q.enqueue(50);
    q.display();
    q.dequeue();
    q.display();
    // Some more test cases with different values
    que q2(5);
    q2.enqueue(15);
    q2.enqueue(25);
    q2.enqueue(35);
    q2.enqueue(45);
    q2.enqueue(55);
    q2.display();
    q2.dequeue();
    q2.display();

    // testing shift Left
    que q3(5);
    q3.enqueue(15);
    q3.enqueue(25);
    q3.enqueue(35);
    q3.enqueue(45);
    q3.enqueue(55);
    q3.display();
          return 0;
}
```

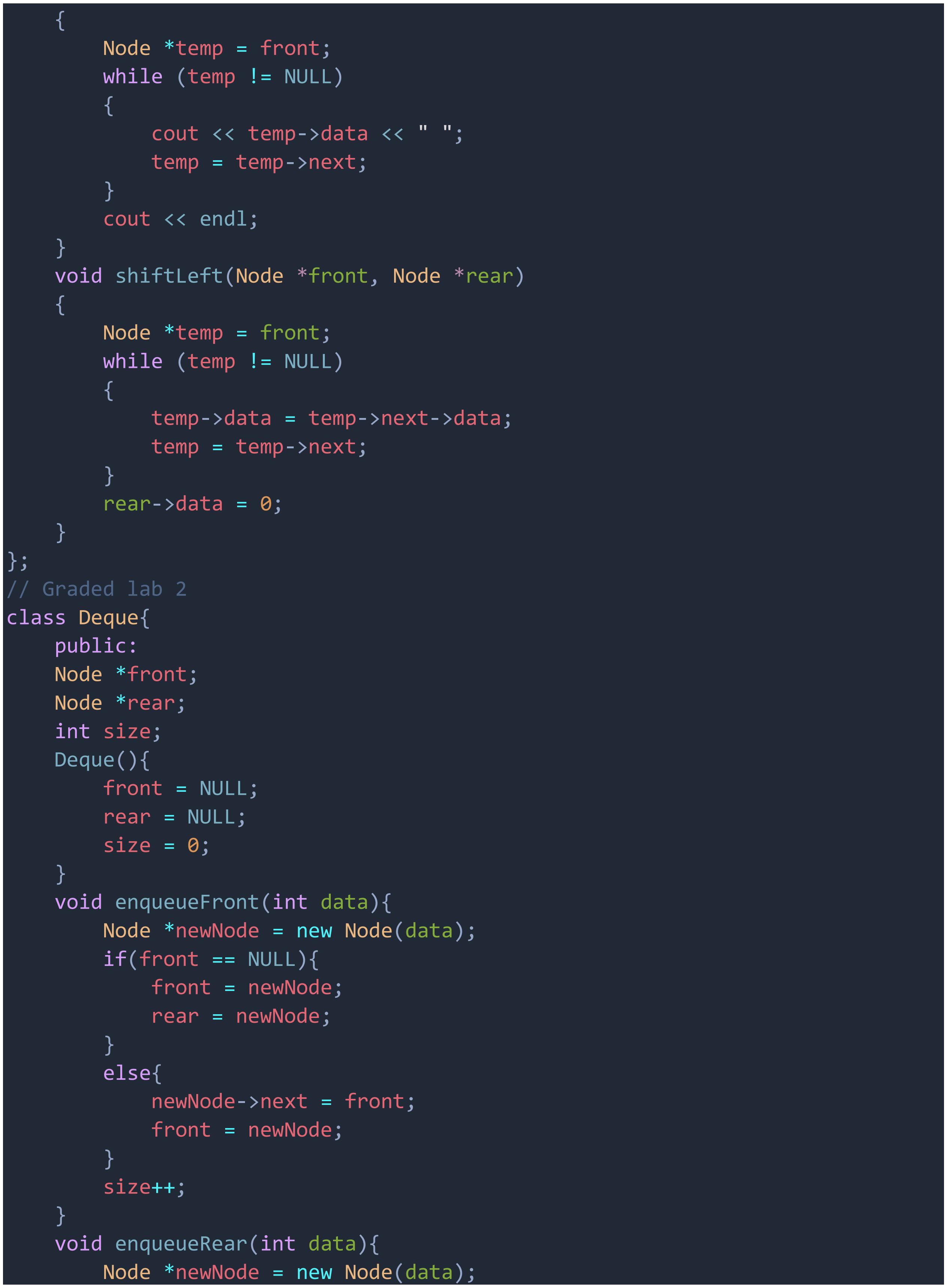
Output:



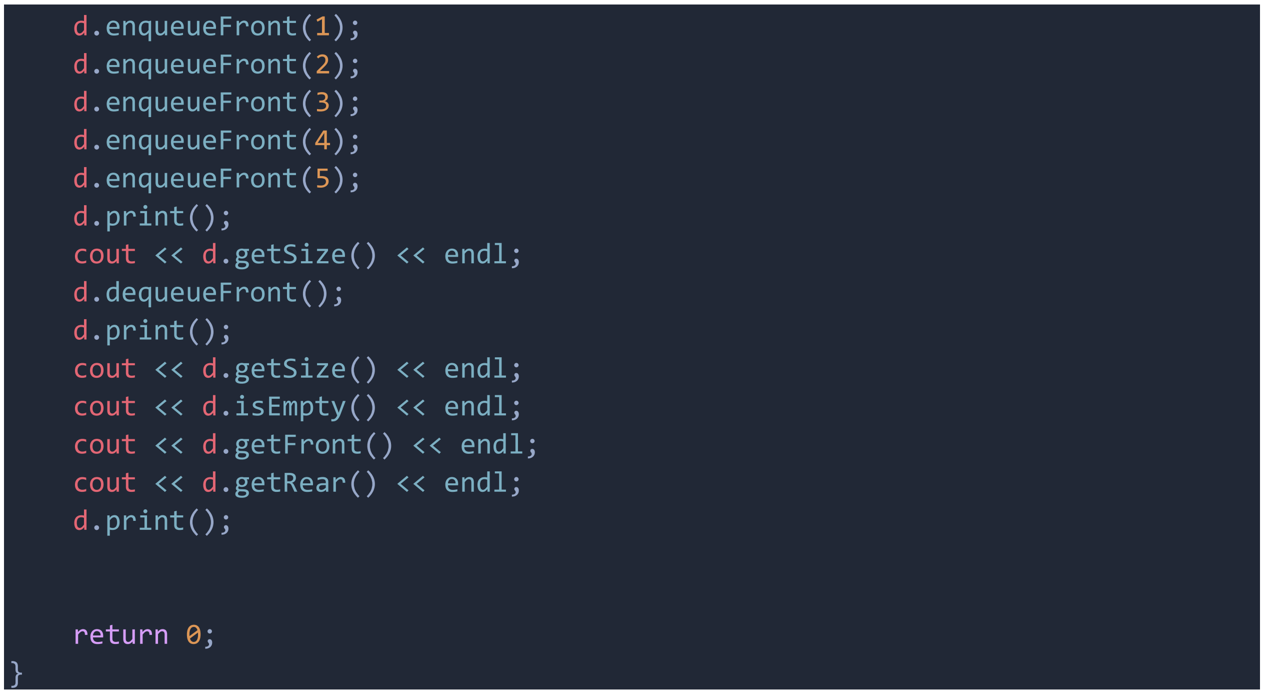
```
PS D:\Ishtudy Material\3rd Sem\DSA\LAB\
10 20 30 40 50
20 30 40 50
15 25 35 45 55
25 35 45 55
15 25 35 45 55
PS D:\Ishtudy Material\3rd Sem\DSA\LAB\
```

Dynamic Queue and Deque:

```cpp
#include <iostream>
using namespace std;
// Graded Lab 1

/*
Implement the methods developed in Activity 1 for Dynmaic Queue i.e. Linked
Implementation of the Queue.
*/

class Node
{
public:
    int data;
    Node *next;
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }
};
class Queue
{
public:
    Node *front;
    Node *rear;
    int size;
    Queue()
    {
        front = NULL;
        rear = NULL;
        size = 0;
    }
    void enqueue(int data)
    {
        Node *newNode = new Node(data);
        if (front == NULL)
        {
            front = newNode;
            rear = newNode;
        }
        else
        {
            rear->next = newNode;
            rear = newNode;
        }
        size++;
    }
    void dequeue()
    {
        if (front == NULL)
        {
            cout << "Queue is empty" << endl;
        }
        else
        {
            Node *temp = front;
            front = front->next;
            delete temp;
            size--;
        }
    }
    int getSize()
    {
        return size;
    }
    bool isEmpty()
    {
        return size == 0;
    }
    int getFront()
    {
        if (front == NULL)
        {
            cout << "Queue is empty" << endl;
            return -1;
        }
        return front->data;
    }
    int getRear()
    {
        if (rear == NULL)
        {
            cout << "Queue is empty" << endl;
            return -1;
        }
        return rear->data;
    }
    void print()
```

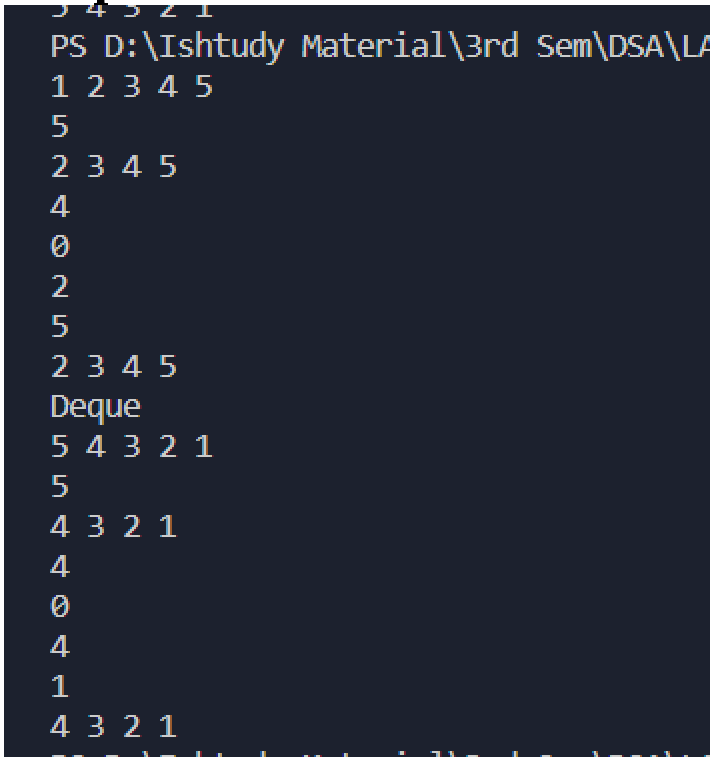


```
    {
        Node *temp = front;
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
    void shiftLeft(Node *front, Node *rear)
    {
        Node *temp = front;
        while (temp != NULL)
        {
            temp->data = temp->next->data;
            temp = temp->next;
        }
        rear->data = 0;
    }
};
// Graded lab 2
class Deque{
    public:
    Node *front;
    Node *rear;
    int size;
    Deque(){
        front = NULL;
        rear = NULL;
        size = 0;
    }
    void enqueueFront(int data){
        Node *newNode = new Node(data);
        if(front == NULL){
            front = newNode;
            rear = newNode;
        }
        else{
            newNode->next = front;
            front = newNode;
        }
        size++;
    }
    void enqueueRear(int data){
        Node *newNode = new Node(data);
        if(front == NULL){
            front = newNode;
            rear = newNode;
        }
        else{
            rear->next = newNode;
            rear = newNode;
        }
        size++;
    }
    void dequeueFront(){
        if(front == NULL){
            cout << "Queue is empty" << endl;
        }
        else{
            Node *temp = front;
            front = front->next;
            delete temp;
            size--;
        }
    }
    void dequeueRear(){
        if(front == NULL){
            cout << "Queue is empty" << endl;
        }
        else{
            Node *temp = front;
            while(temp->next != rear){
                temp = temp->next;
            }
            delete rear;
            rear = temp;
            rear->next = NULL;
            size--;
        }
    }
    int getSize(){
        return size;
    }
    bool isEmpty(){
        return size == 0;
    }
    int getFront(){
        if(front == NULL){
            cout << "Queue is empty" << endl;
            return -1;
        }
        return front->data;
    }
    int getRear(){
        if(rear == NULL){
            cout << "Queue is empty" << endl;
            return -1;
        }
        return rear->data;
    }
    void print(){
        Node *temp = front;
        while(temp != NULL){
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main()
{
    // Implimenting Queue
    Queue q;
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    q.enqueue(4);
    q.enqueue(5);
    q.print();
    cout << q.getSize() << endl;
    q.dequeue();
    q.print();
    cout << q.getSize() << endl;
    cout << q.isEmpty() << endl;
    cout << q.getFront() << endl;
    cout << q.getRear() << endl;
    q.print();

    // Implimenting Deque

    cout<<"Deque" << endl;

    Deque d;
```



```cpp
    d.enqueueFront(1);
    d.enqueueFront(2);
    d.enqueueFront(3);
    d.enqueueFront(4);
    d.enqueueFront(5);
    d.print();
    cout << d.getSize() << endl;
    d.dequeueFront();
    d.print();
    cout << d.getSize() << endl;
    cout << d.isEmpty() << endl;
    cout << d.getFront() << endl;
    cout << d.getRear() << endl;
    d.print();


    return 0;
}
```

Output:



```
5 4 3 2 1
PS D:\Ishtudy Material\3rd Sem\DSA\L
1 2 3 4 5
5
2 3 4 5
4
0
2
5
2 3 4 5
Deque
5 4 3 2 1
5
4 3 2 1
4
0
4
1
4 3 2 1
```