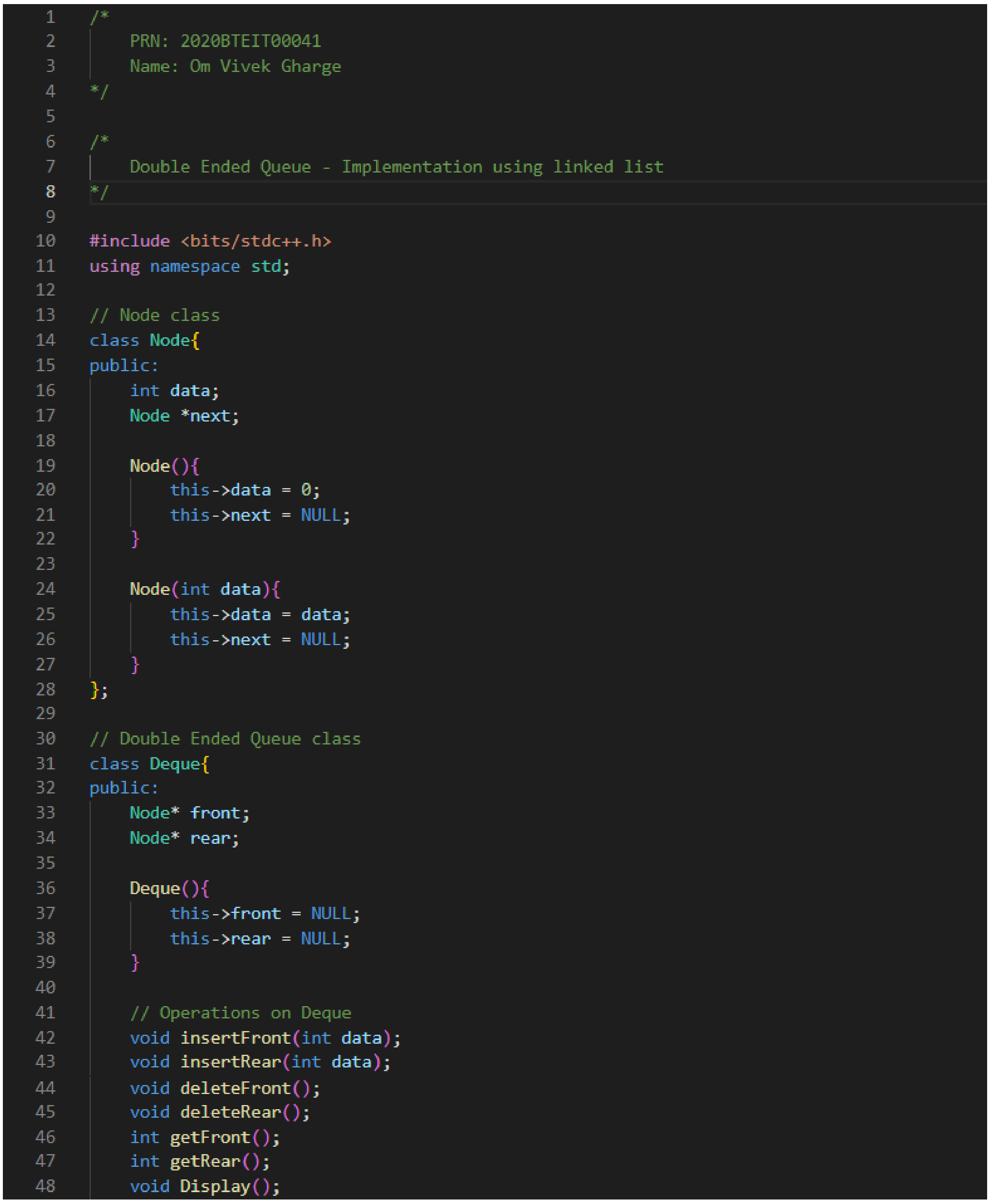
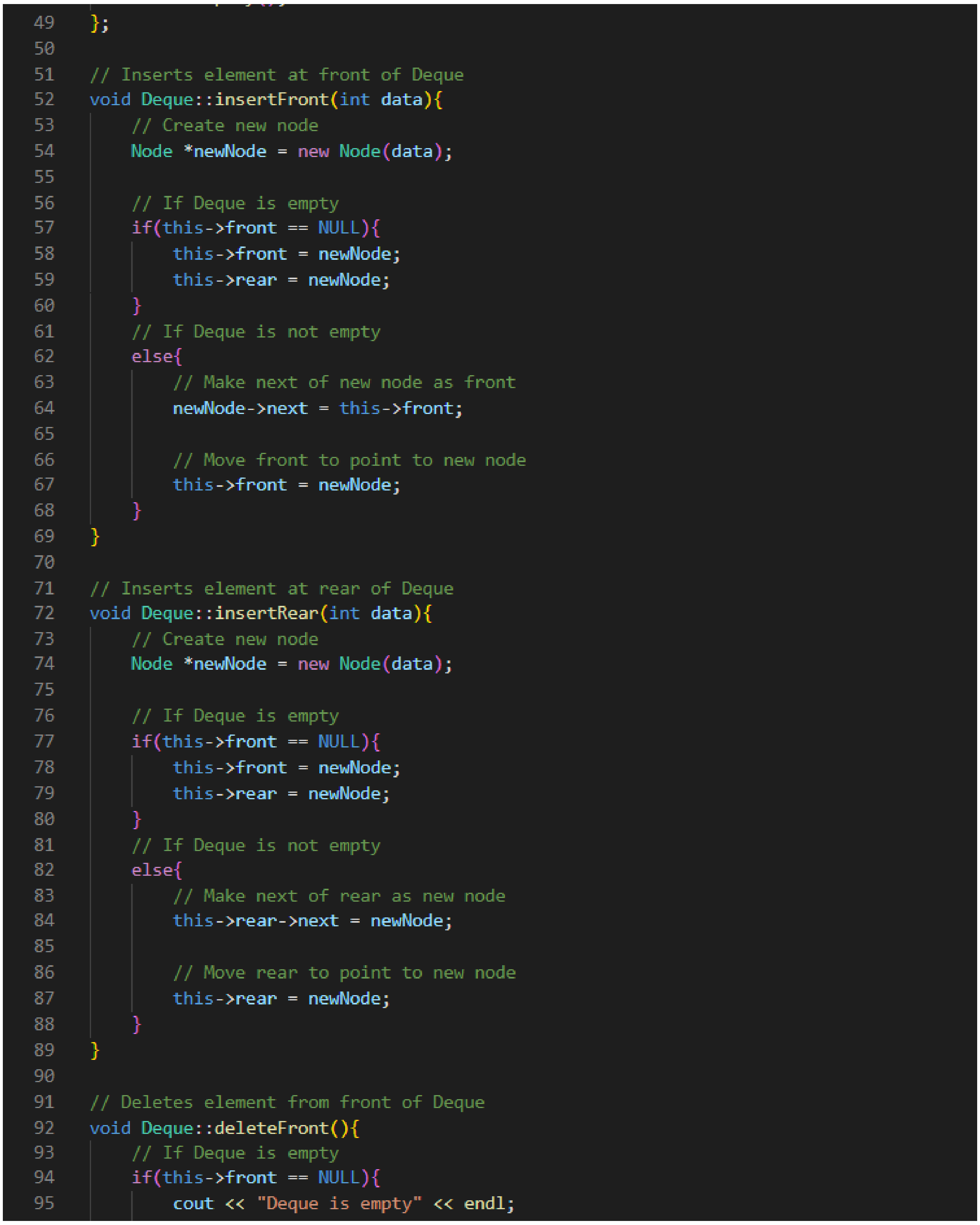
PRN: 2020BTEIT00041

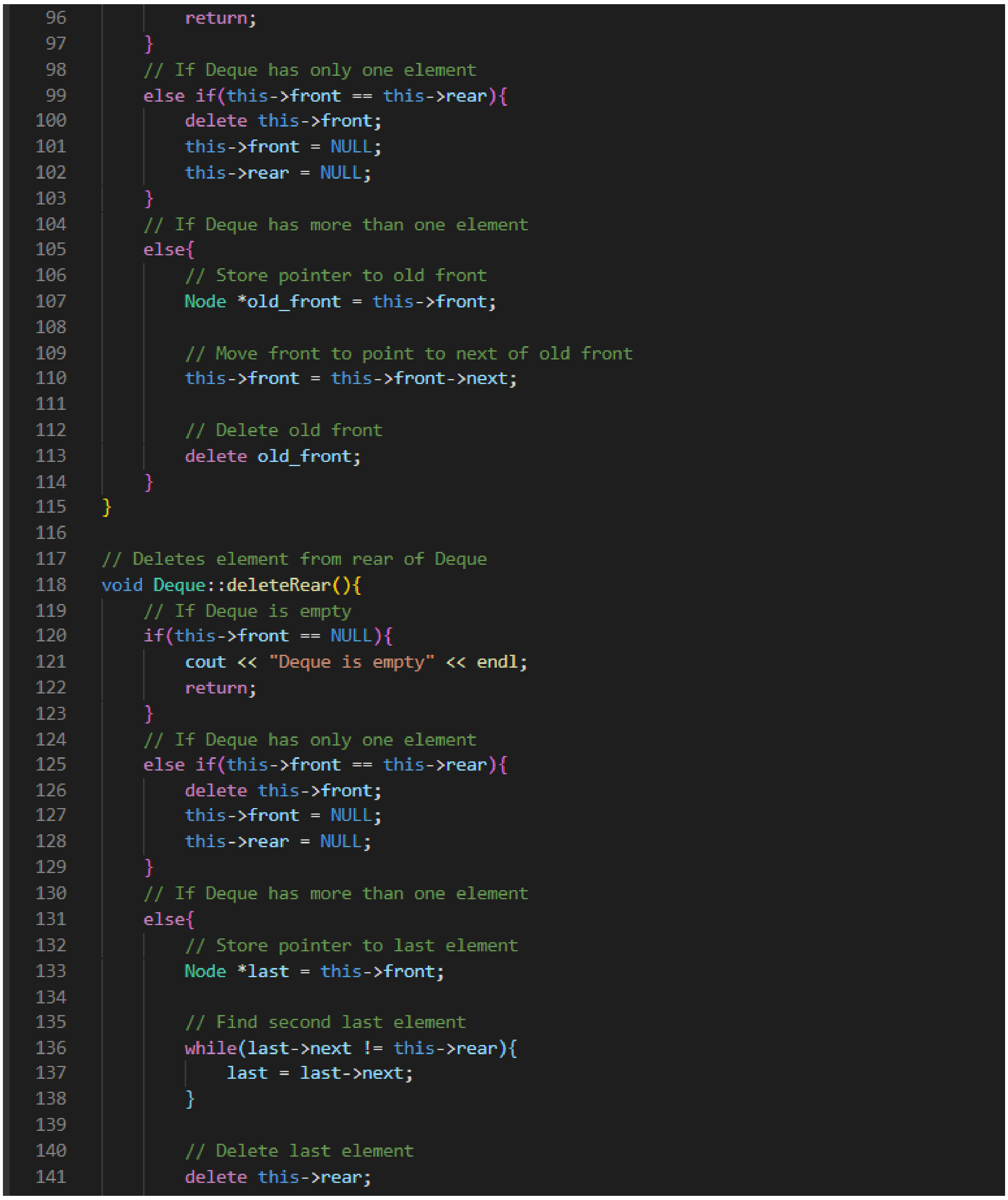
Double Ended Queue:



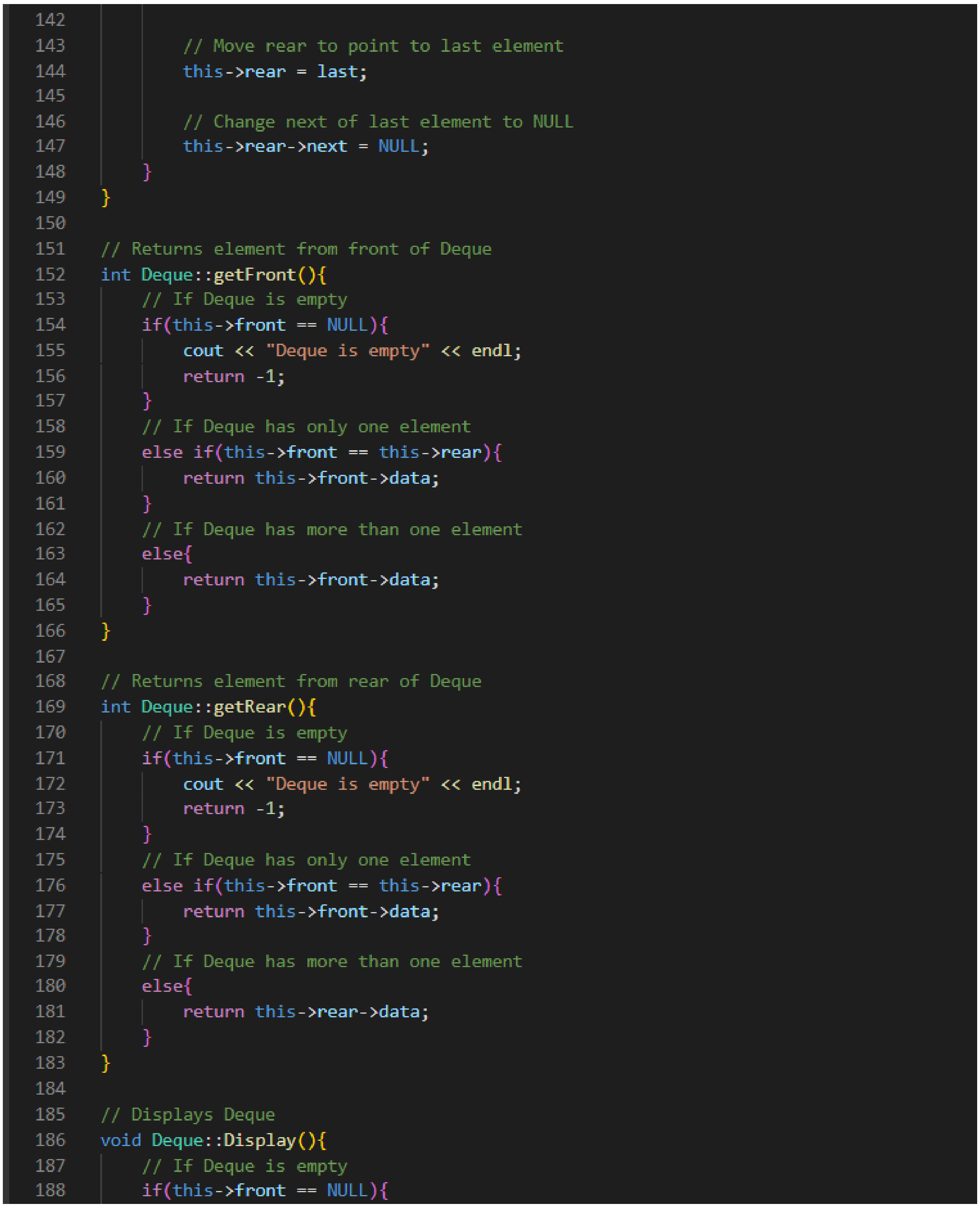
```cpp
/*
    PRN: 2020BTEIT00041
    Name: Om Vivek Gharge
*/

/*
    Double Ended Queue - Implementation using linked list
*/

#include <bits/stdc++.h>
using namespace std;

// Node class
class Node{
public:
    int data;
    Node *next;

    Node(){
        this->data = 0;
        this->next = NULL;
    }

    Node(int data){
        this->data = data;
        this->next = NULL;
    }
};

// Double Ended Queue class
class Deque{
public:
    Node* front;
    Node* rear;

    Deque(){
        this->front = NULL;
        this->rear = NULL;
    }

    // Operations on Deque
    void insertFront(int data);
    void insertRear(int data);
    void deleteFront();
    void deleteRear();
    int getFront();
    int getRear();
    void Display();
```



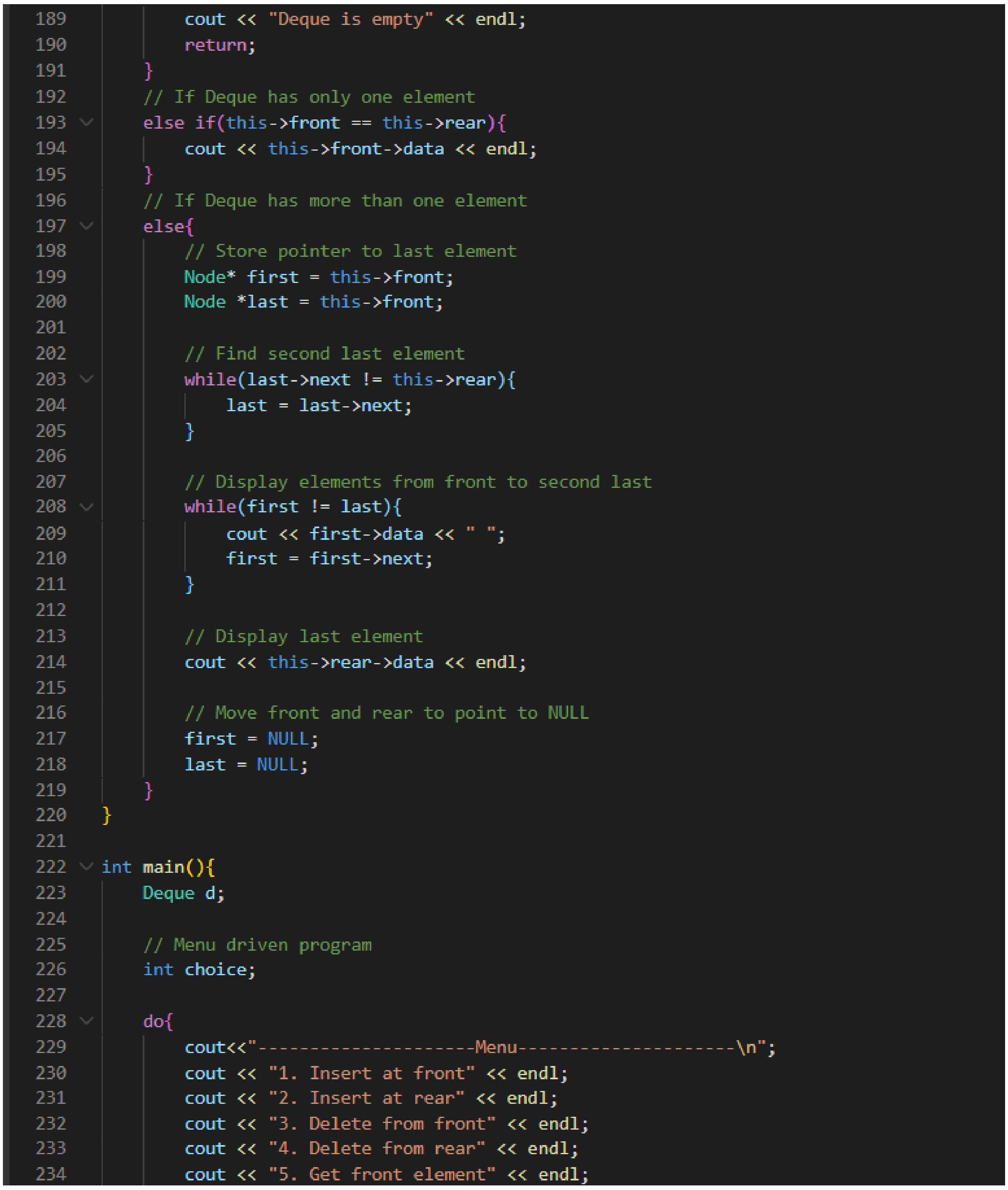
```cpp
};

// Inserts element at front of Deque
void Deque::insertFront(int data){
    // Create new node
    Node *newNode = new Node(data);

    // If Deque is empty
    if(this->front == NULL){
        this->front = newNode;
        this->rear = newNode;
    }
    // If Deque is not empty
    else{
        // Make next of new node as front
        newNode->next = this->front;

        // Move front to point to new node
        this->front = newNode;
    }
}

// Inserts element at rear of Deque
void Deque::insertRear(int data){
    // Create new node
    Node *newNode = new Node(data);

    // If Deque is empty
    if(this->front == NULL){
        this->front = newNode;
        this->rear = newNode;
    }
    // If Deque is not empty
    else{
        // Make next of rear as new node
        this->rear->next = newNode;

        // Move rear to point to new node
        this->rear = newNode;
    }
}

// Deletes element from front of Deque
void Deque::deleteFront(){
    // If Deque is empty
    if(this->front == NULL){
        cout << "Deque is empty" << endl;
```



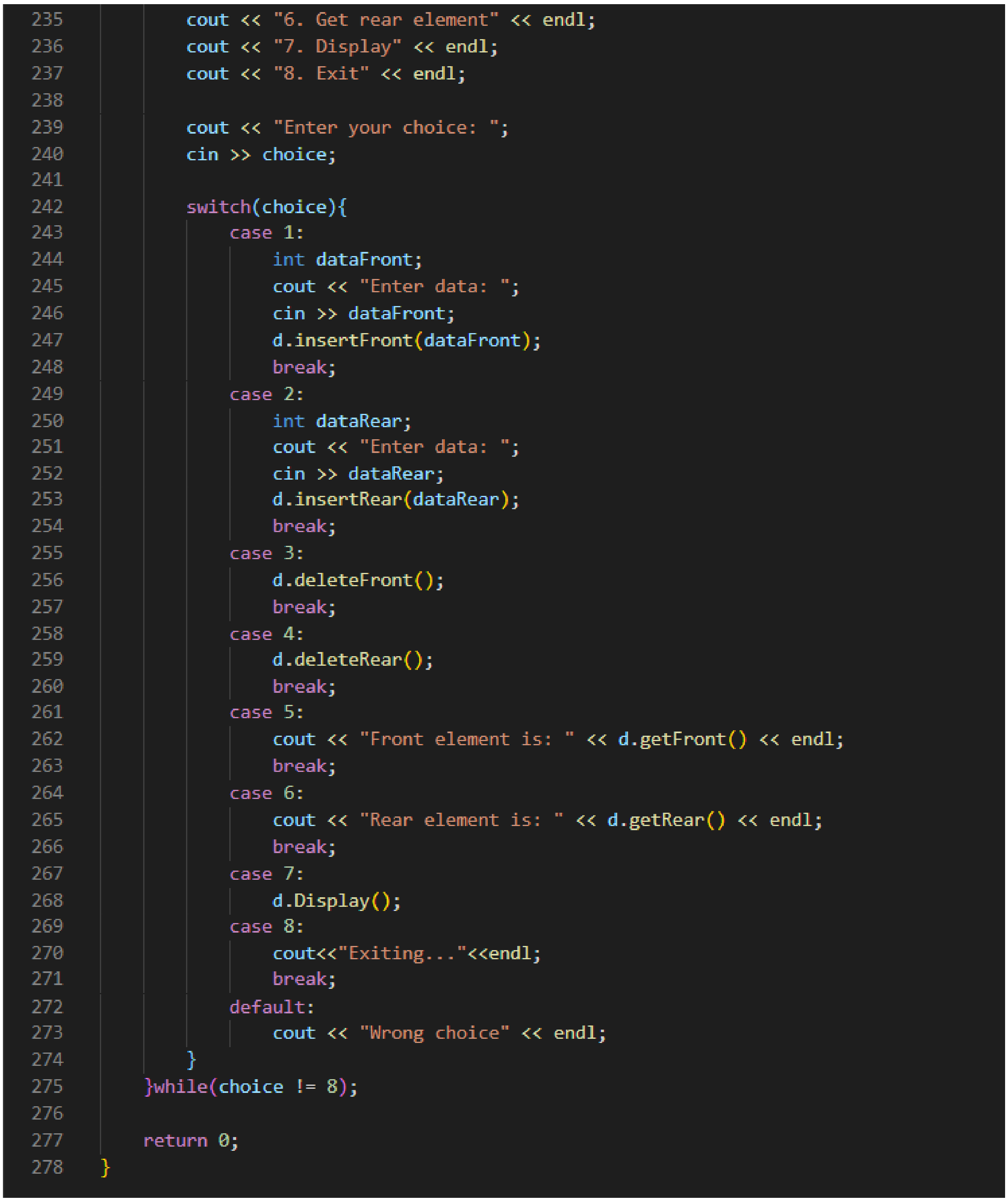
```
        return;
    }
    // If Deque has only one element
    else if(this->front == this->rear){
        delete this->front;
        this->front = NULL;
        this->rear = NULL;
    }
    // If Deque has more than one element
    else{
        // Store pointer to old front
        Node *old_front = this->front;

        // Move front to point to next of old front
        this->front = this->front->next;

        // Delete old front
        delete old_front;
    }
}

// Deletes element from rear of Deque
void Deque::deleteRear(){
    // If Deque is empty
    if(this->front == NULL){
        cout << "Deque is empty" << endl;
        return;
    }
    // If Deque has only one element
    else if(this->front == this->rear){
        delete this->front;
        this->front = NULL;
        this->rear = NULL;
    }
    // If Deque has more than one element
    else{
        // Store pointer to last element
        Node *last = this->front;

        // Find second last element
        while(last->next != this->rear){
            last = last->next;
        }

        // Delete last element
        delete this->rear;
```



```cpp

        // Move rear to point to last element
        this->rear = last;

        // Change next of last element to NULL
        this->rear->next = NULL;
    }
}

// Returns element from front of Deque
int Deque::getFront(){
    // If Deque is empty
    if(this->front == NULL){
        cout << "Deque is empty" << endl;
        return -1;
    }
    // If Deque has only one element
    else if(this->front == this->rear){
        return this->front->data;
    }
    // If Deque has more than one element
    else{
        return this->front->data;
    }
}

// Returns element from rear of Deque
int Deque::getRear(){
    // If Deque is empty
    if(this->front == NULL){
        cout << "Deque is empty" << endl;
        return -1;
    }
    // If Deque has only one element
    else if(this->front == this->rear){
        return this->front->data;
    }
    // If Deque has more than one element
    else{
        return this->rear->data;
    }
}

// Displays Deque
void Deque::Display(){
    // If Deque is empty
    if(this->front == NULL){
```

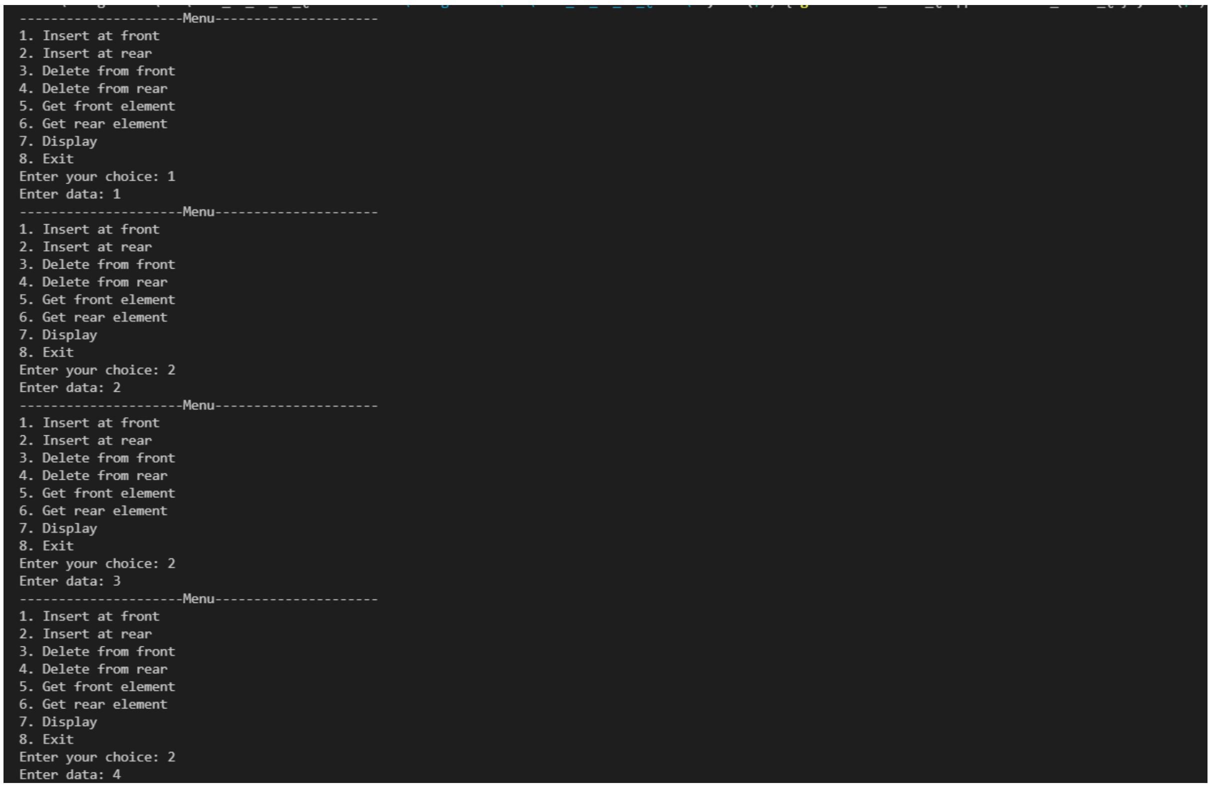


```cpp
        cout << "Deque is empty" << endl;
        return;
    }
    // If Deque has only one element
    else if(this->front == this->rear){
        cout << this->front->data << endl;
    }
    // If Deque has more than one element
    else{
        // Store pointer to last element
        Node* first = this->front;
        Node *last = this->front;

        // Find second last element
        while(last->next != this->rear){
            last = last->next;
        }

        // Display elements from front to second last
        while(first != last){
            cout << first->data << " ";
            first = first->next;
        }

        // Display last element
        cout << this->rear->data << endl;

        // Move front and rear to point to NULL
        first = NULL;
        last = NULL;
    }
}

int main(){
    Deque d;

    // Menu driven program
    int choice;

    do{
        cout<<"--------------------Menu--------------------\n";
        cout << "1. Insert at front" << endl;
        cout << "2. Insert at rear" << endl;
        cout << "3. Delete from front" << endl;
        cout << "4. Delete from rear" << endl;
        cout << "5. Get front element" << endl;
```

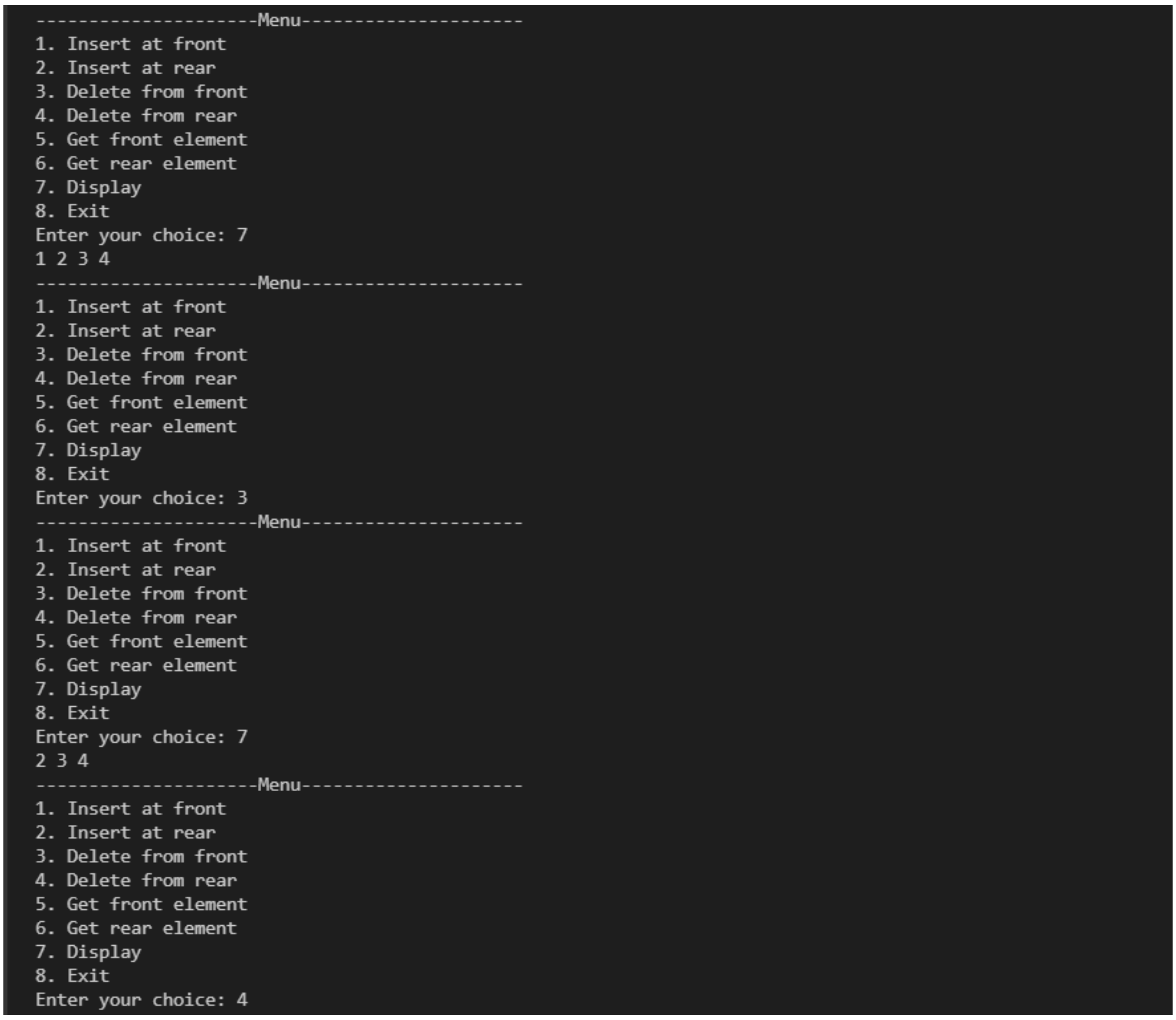


```cpp
        cout << "6. Get rear element" << endl;
        cout << "7. Display" << endl;
        cout << "8. Exit" << endl;

        cout << "Enter your choice: ";
        cin >> choice;

        switch(choice){
            case 1:
                int dataFront;
                cout << "Enter data: ";
                cin >> dataFront;
                d.insertFront(dataFront);
                break;
            case 2:
                int dataRear;
                cout << "Enter data: ";
                cin >> dataRear;
                d.insertRear(dataRear);
                break;
            case 3:
                d.deleteFront();
                break;
            case 4:
                d.deleteRear();
                break;
            case 5:
                cout << "Front element is: " << d.getFront() << endl;
                break;
            case 6:
                cout << "Rear element is: " << d.getRear() << endl;
                break;
            case 7:
                d.Display();
            case 8:
                cout<<"Exiting..."<<endl;
                break;
            default:
                cout << "Wrong choice" << endl;
        }
    }while(choice != 8);

    return 0;
}
```

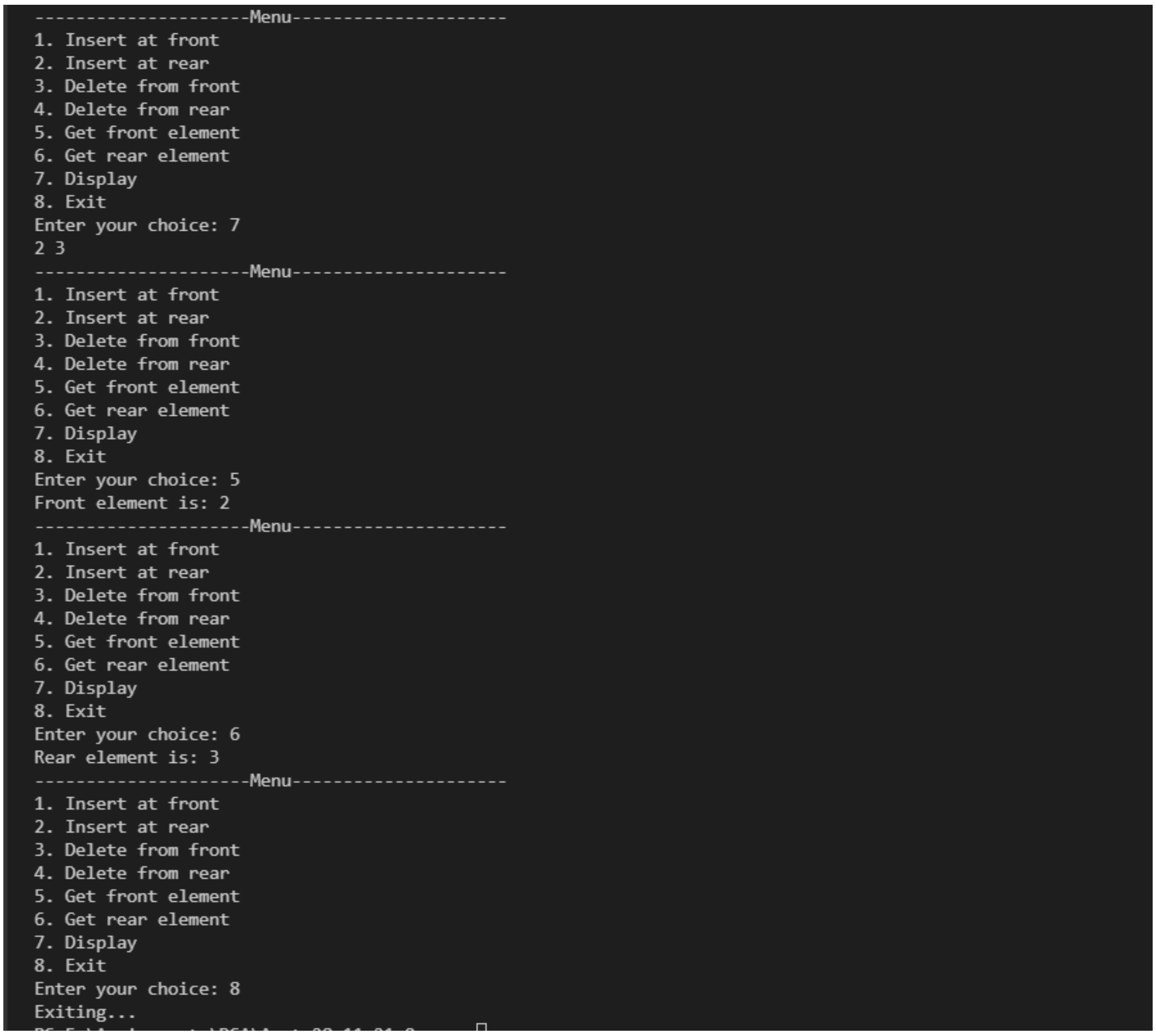
OUTPUT:



```
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 1
Enter data: 1
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 2
Enter data: 2
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 2
Enter data: 3
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 2
Enter data: 4
```



```
---------------------Menu---------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 7
1 2 3 4
---------------------Menu---------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 3
---------------------Menu---------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 7
2 3 4
---------------------Menu---------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 4
```



```
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 7
2 3
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 5
Front element is: 2
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 6
Rear element is: 3
--------------------Menu--------------------
1. Insert at front
2. Insert at rear
3. Delete from front
4. Delete from rear
5. Get front element
6. Get rear element
7. Display
8. Exit
Enter your choice: 8
Exiting...
[illegible]
```