Data Structure: Queue

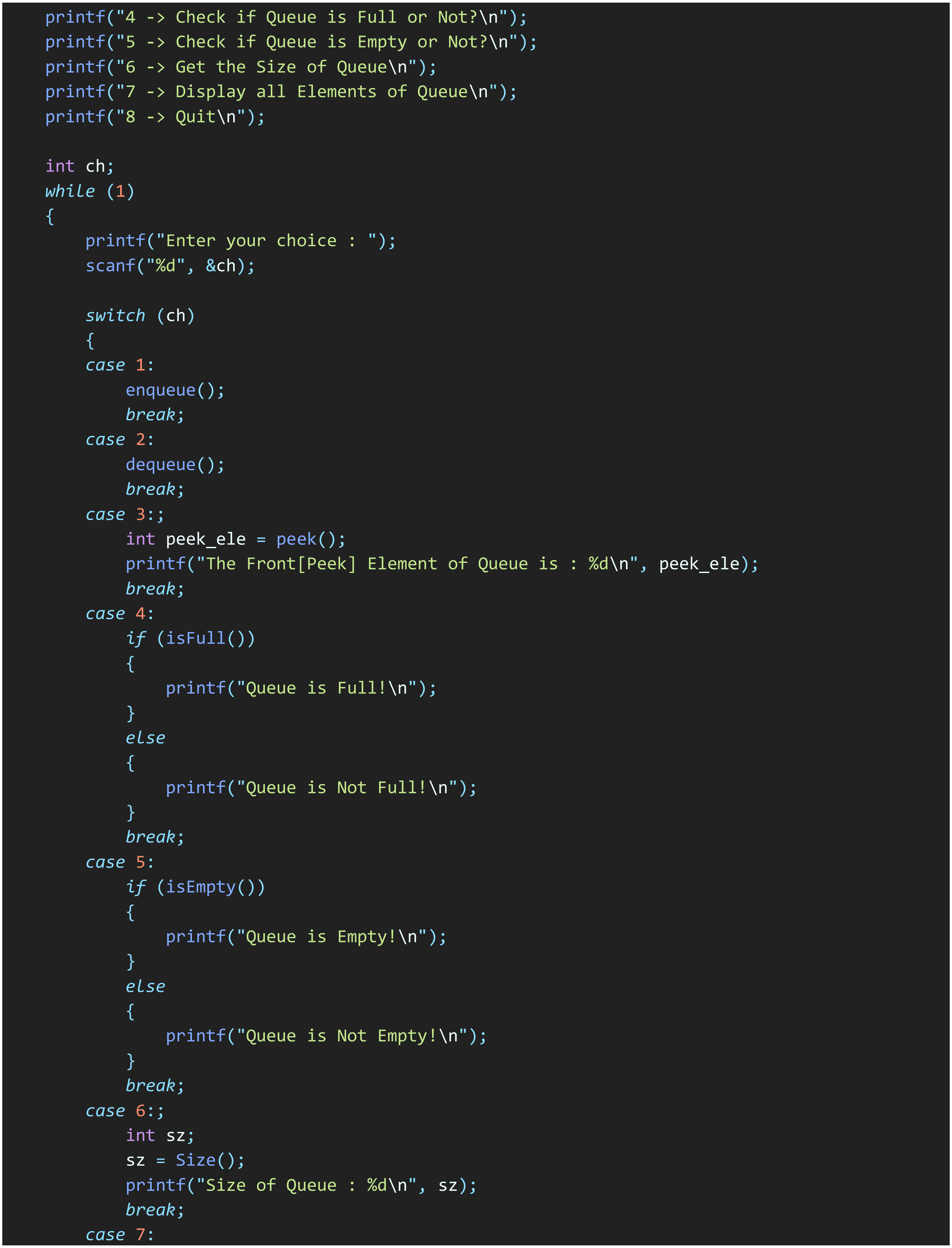
Tutorial 6: U19CS012

1. Write a program to implement a queue and perform basic operations of queue using array.

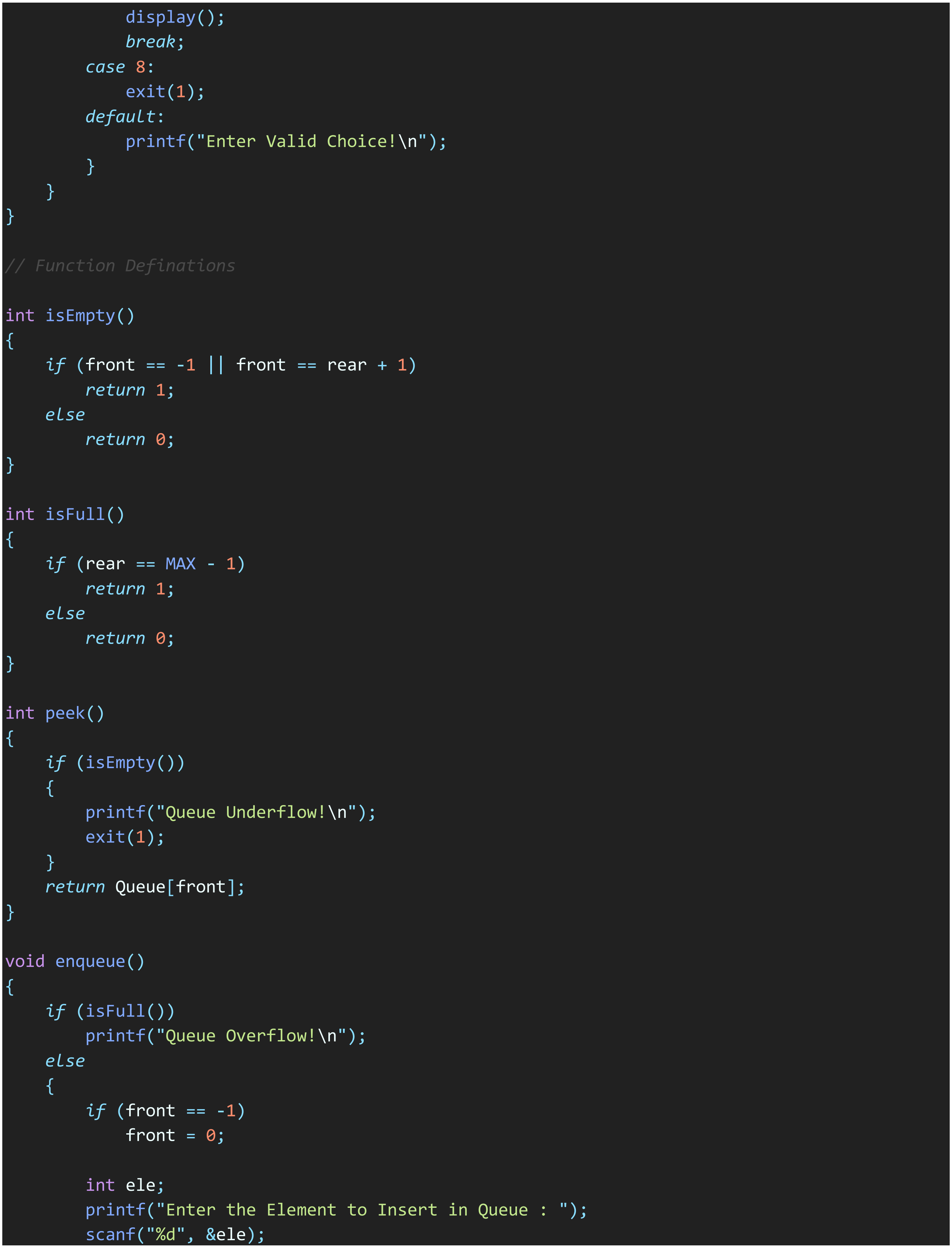
- En(enter)queue
- De(lete)queue
- peek
- isfull
- isempty

Code:

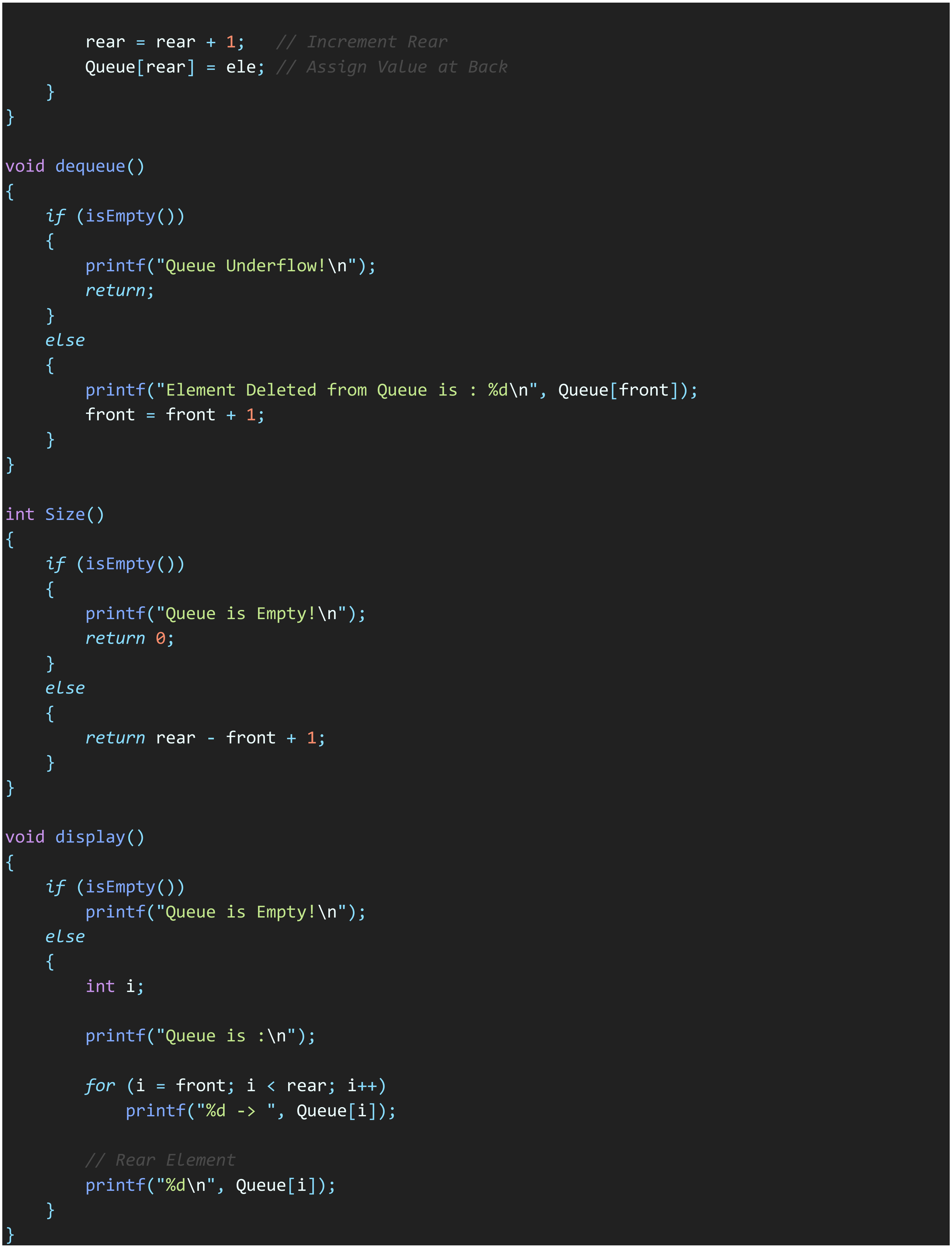
```
// Write a program to implement a queue and perform basic operations of queue using array.
// •    En(enter)queue
// •    De(lete)queue
// •    peek
// •    isfull
// •    isempty
// •    size
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

// Function Declarations
void enqueue();
void dequeue();

//Supportive Operations
int peek();
int isFull();
int isEmpty();
int Size();
void display();

int Queue[MAX];

//Initially Front and Rear Location is -1
int front = -1;
int rear = -1;

int main()
{
    // Menu Driven C Program for Queue implementation Using Array
    printf("\n~~~~~~Queue Operations~~~~~~\n");

    printf("1 -> Enter Element in Queue\n");
    printf("2 -> Delete Element from Queue\n");
    printf("3 -> Peek[Front] Element of Queue\n");
```



```
    printf("4 -> Check if Queue is Full or Not?\n");
    printf("5 -> Check if Queue is Empty or Not?\n");
    printf("6 -> Get the Size of Queue\n");
    printf("7 -> Display all Elements of Queue\n");
    printf("8 -> Quit\n");

    int ch;
    while (1)
    {
        printf("Enter your choice : ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            enqueue();
            break;
        case 2:
            dequeue();
            break;
        case 3:;
            int peek_ele = peek();
            printf("The Front[Peek] Element of Queue is : %d\n", peek_ele);
            break;
        case 4:
            if (isFull())
            {
                printf("Queue is Full!\n");
            }
            else
            {
                printf("Queue is Not Full!\n");
            }
            break;
        case 5:
            if (isEmpty())
            {
                printf("Queue is Empty!\n");
            }
            else
            {
                printf("Queue is Not Empty!\n");
            }
            break;
        case 6:;
            int sz;
            sz = Size();
            printf("Size of Queue : %d\n", sz);
            break;
        case 7:
```

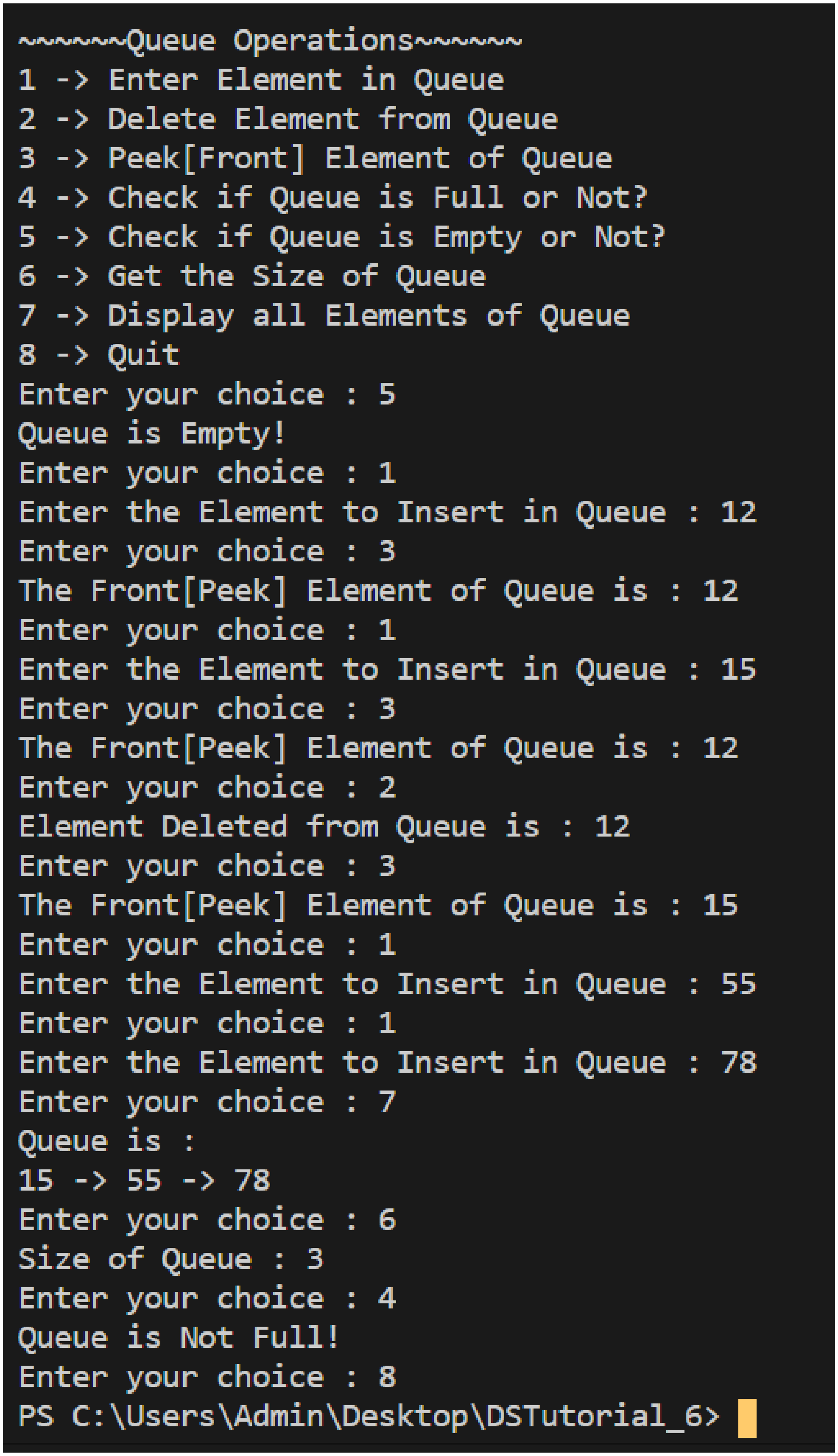


```
            display();
            break;
        case 8:
            exit(1);
        default:
            printf("Enter Valid Choice!\n");
        }
    }
}

// Function Definations

int isEmpty()
{
    if (front == -1 || front == rear + 1)
        return 1;
    else
        return 0;
}

int isFull()
{
    if (rear == MAX - 1)
        return 1;
    else
        return 0;
}

int peek()
{
    if (isEmpty())
    {
        printf("Queue Underflow!\n");
        exit(1);
    }
    return Queue[front];
}

void enqueue()
{
    if (isFull())
        printf("Queue Overflow!\n");
    else
    {
        if (front == -1)
            front = 0;

        int ele;
        printf("Enter the Element to Insert in Queue : ");
        scanf("%d", &ele);
```



```c
        rear = rear + 1;   // Increment Rear
        Queue[rear] = ele; // Assign Value at Back
    }
}

void dequeue()
{
    if (isEmpty())
    {
        printf("Queue Underflow!\n");
        return;
    }
    else
    {
        printf("Element Deleted from Queue is : %d\n", Queue[front]);
        front = front + 1;
    }
}

int Size()
{
    if (isEmpty())
    {
        printf("Queue is Empty!\n");
        return 0;
    }
    else
    {
        return rear - front + 1;
    }
}

void display()
{
    if (isEmpty())
        printf("Queue is Empty!\n");
    else
    {
        int i;

        printf("Queue is :\n");

        for (i = front; i < rear; i++)
            printf("%d -> ", Queue[i]);

        // Rear Element
        printf("%d\n", Queue[i]);
    }
}
```

Example Test Cases:

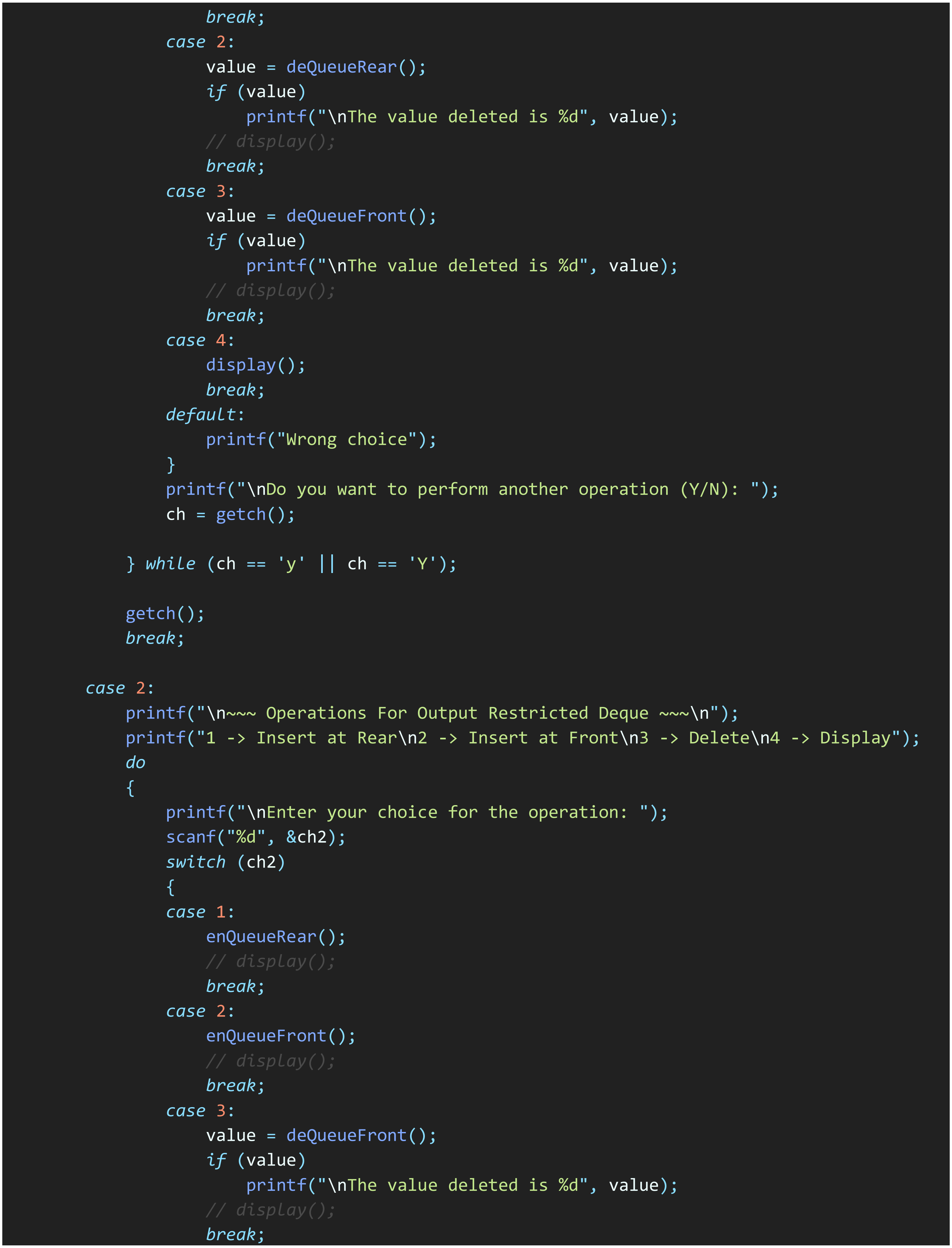


```
~~~~~~Queue Operations~~~~~~
1 -> Enter Element in Queue
2 -> Delete Element from Queue
3 -> Peek[Front] Element of Queue
4 -> Check if Queue is Full or Not?
5 -> Check if Queue is Empty or Not?
6 -> Get the Size of Queue
7 -> Display all Elements of Queue
8 -> Quit
Enter your choice : 5
Queue is Empty!
Enter your choice : 1
Enter the Element to Insert in Queue : 12
Enter your choice : 3
The Front[Peek] Element of Queue is : 12
Enter your choice : 1
Enter the Element to Insert in Queue : 15
Enter your choice : 3
The Front[Peek] Element of Queue is : 12
Enter your choice : 2
Element Deleted from Queue is : 12
Enter your choice : 3
The Front[Peek] Element of Queue is : 15
Enter your choice : 1
Enter the Element to Insert in Queue : 55
Enter your choice : 1
Enter the Element to Insert in Queue : 78
Enter your choice : 7
Queue is :
15 -> 55 -> 78
Enter your choice : 6
Size of Queue : 3
Enter your choice : 4
Queue is Not Full!
Enter your choice : 8
PS C:\Users\Admin\Desktop\DSTutorial_6>
```

2. Perform above operations to implement DEQueue (double ended queue)

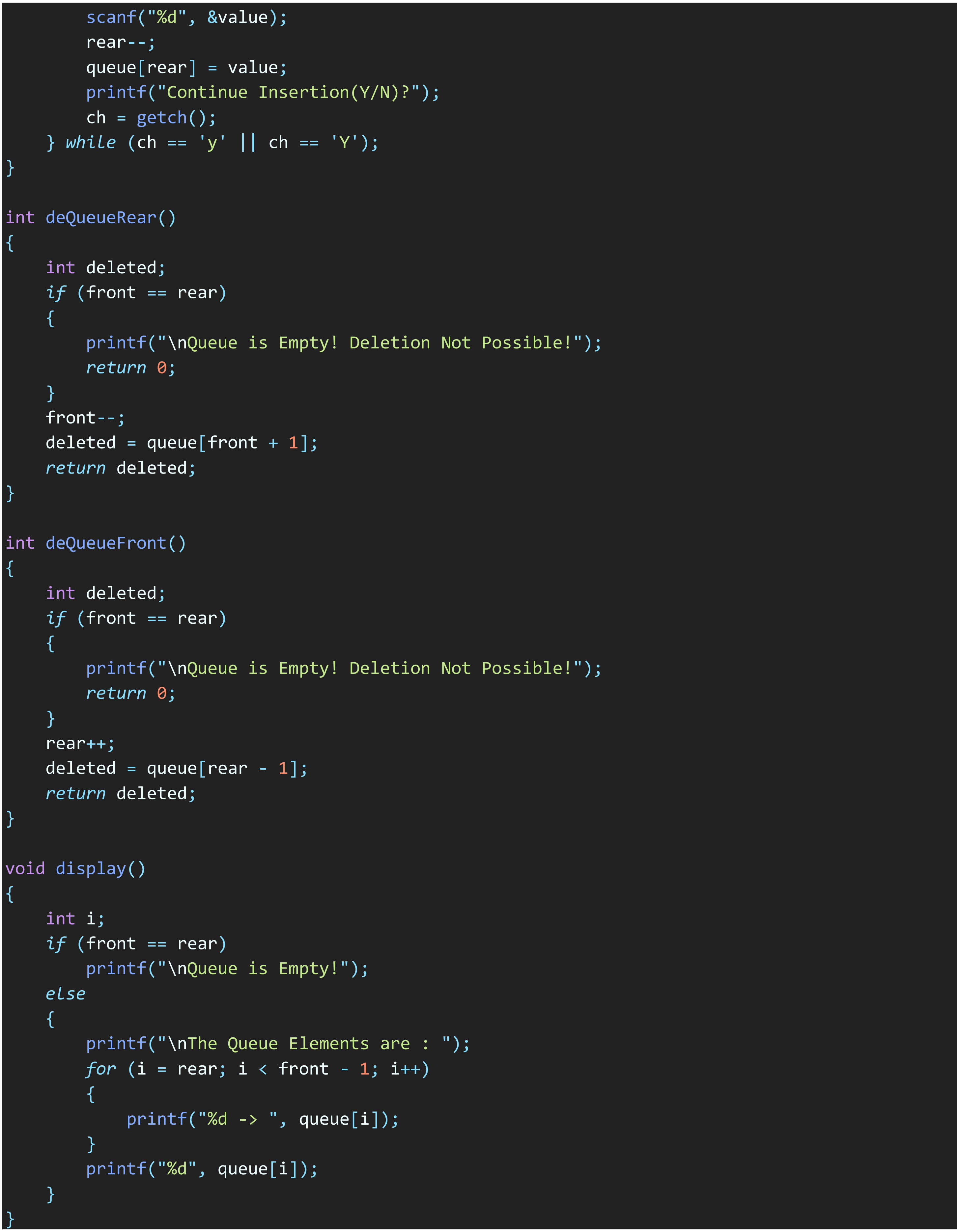
Code:

```
#include <stdio.h>
#include <conio.h>
#define SIZE 100

// Function Defination
void enQueue();
// For INPUT Restricted Queue
int deQueueFront();
int deQueueRear();
// For OUTPUT Restricted Queue
void enQueueRear();
void enQueueFront();

void display();

int queue[SIZE];
int rear = 0, front = 0;

int main()
{
    char ch;
    int ch1, ch2, value = 0;
    printf("\n~~~~~~~ Type of Double Ended Queue ~~~~~~~\n");
    do
    {
        printf("\n1 -> Input-restricted deque \n");
        printf("2 -> Output-restricted deque \n");
        printf("\nEnter your choice of Queue Type : ");
        scanf("%d", &ch1);

        switch (ch1)
        {
        case 1:
            printf("\n~~~ Operations For Input Restricted Deque ~~~\n");

            printf("1 -> Insert\n2 -> Delete from Rear\n3 -> Delete from Front\n4 -
>  Display");

            do
            {
                printf("\nEnter Choice for DeQueue Operation : ");
                scanf("%d", &ch2);
                switch (ch2)
                {
                case 1:
                    enQueueRear();
                    // display();
```



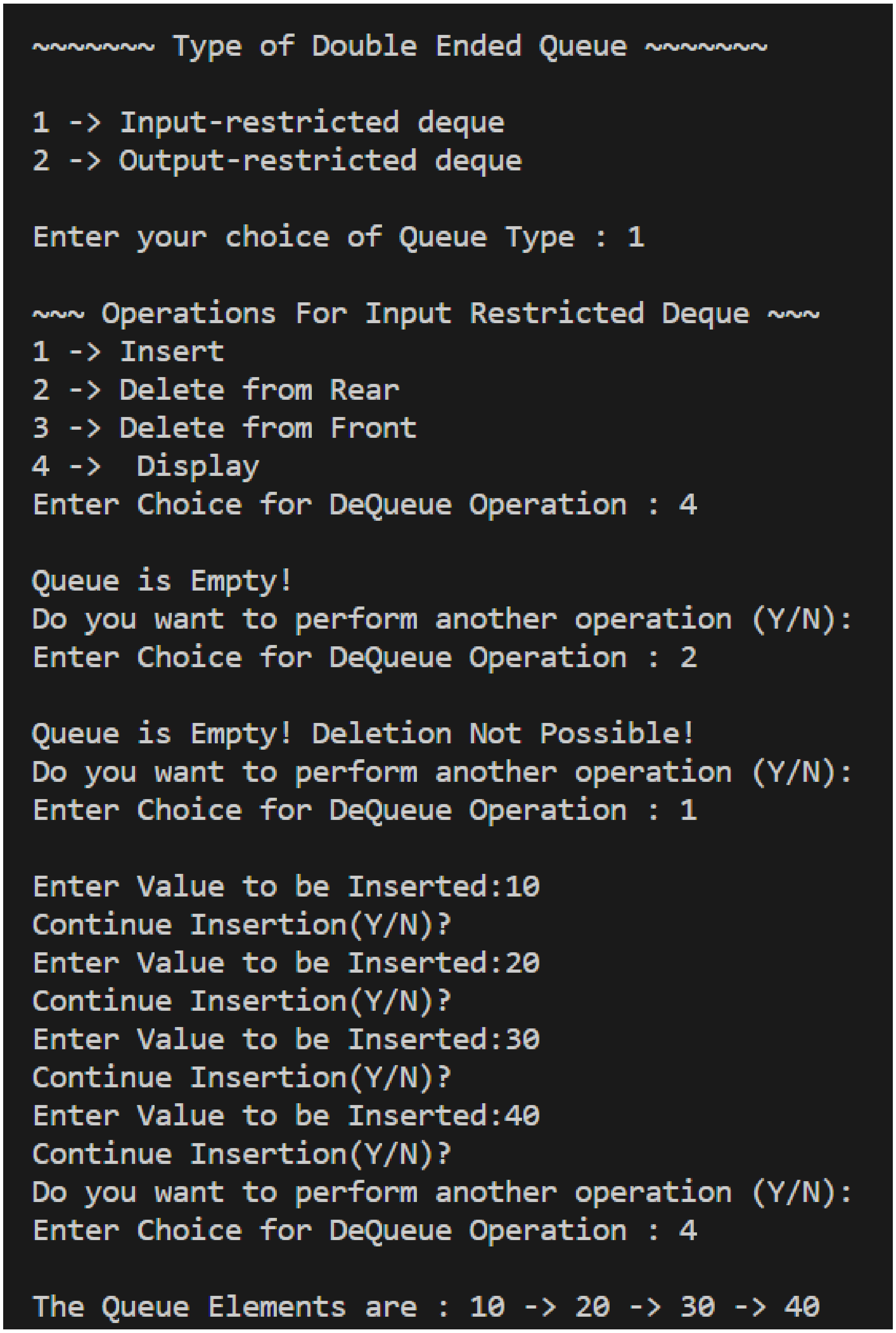
```
                    break;
                case 2:
                    value = deQueueRear();
                    if (value)
                        printf("\nThe value deleted is %d", value);
                    // display();
                    break;
                case 3:
                    value = deQueueFront();
                    if (value)
                        printf("\nThe value deleted is %d", value);
                    // display();
                    break;
                case 4:
                    display();
                    break;
                default:
                    printf("Wrong choice");
                }
                printf("\nDo you want to perform another operation (Y/N): ");
                ch = getch();

            } while (ch == 'y' || ch == 'Y');

            getch();
            break;

        case 2:
            printf("\n~~~ Operations For Output Restricted Deque ~~~\n");
            printf("1 -> Insert at Rear\n2 -> Insert at Front\n3 -> Delete\n4 -> Display");
            do
            {
                printf("\nEnter your choice for the operation: ");
                scanf("%d", &ch2);
                switch (ch2)
                {
                case 1:
                    enQueueRear();
                    // display();
                    break;
                case 2:
                    enQueueFront();
                    // display();
                    break;
                case 3:
                    value = deQueueFront();
                    if (value)
                        printf("\nThe value deleted is %d", value);
                    // display();
                    break;
```

```
                case 4:
                    display();
                    break;
                default:
                    printf("Wrong choice");
                }
                printf("\nDo you want to perform another operation (Y/N): ");
                ch = getch();

            } while (ch == 'y' || ch == 'Y');
            getch();
            break;
        }
        printf("\nDo you want to continue with Another Deque(y/n):");
        ch = getch();
    } while (ch == 'y' || ch == 'Y');
}

void enQueueRear()
{
    char ch;
    if (front == SIZE / 2)
    {
        printf("\nQueue is full!!! Insertion is not possible!!! ");
        return;
    }
    do
    {
        int value;
        printf("\nEnter Value to be Inserted:");
        scanf("%d", &value);
        queue[front] = value;
        front++;
        printf("Continue Insertion(Y/N)?");
        ch = getch();
    } while (ch == 'y' || ch == 'Y');
}

void enQueueFront()
{
    char ch;
    if (front == SIZE / 2)
    {
        printf("\nQueue is full! Insertion Not Possible!");
        return;
    }
    do
    {
        int value;
        printf("\nEnter Value to be Inserted:");
```

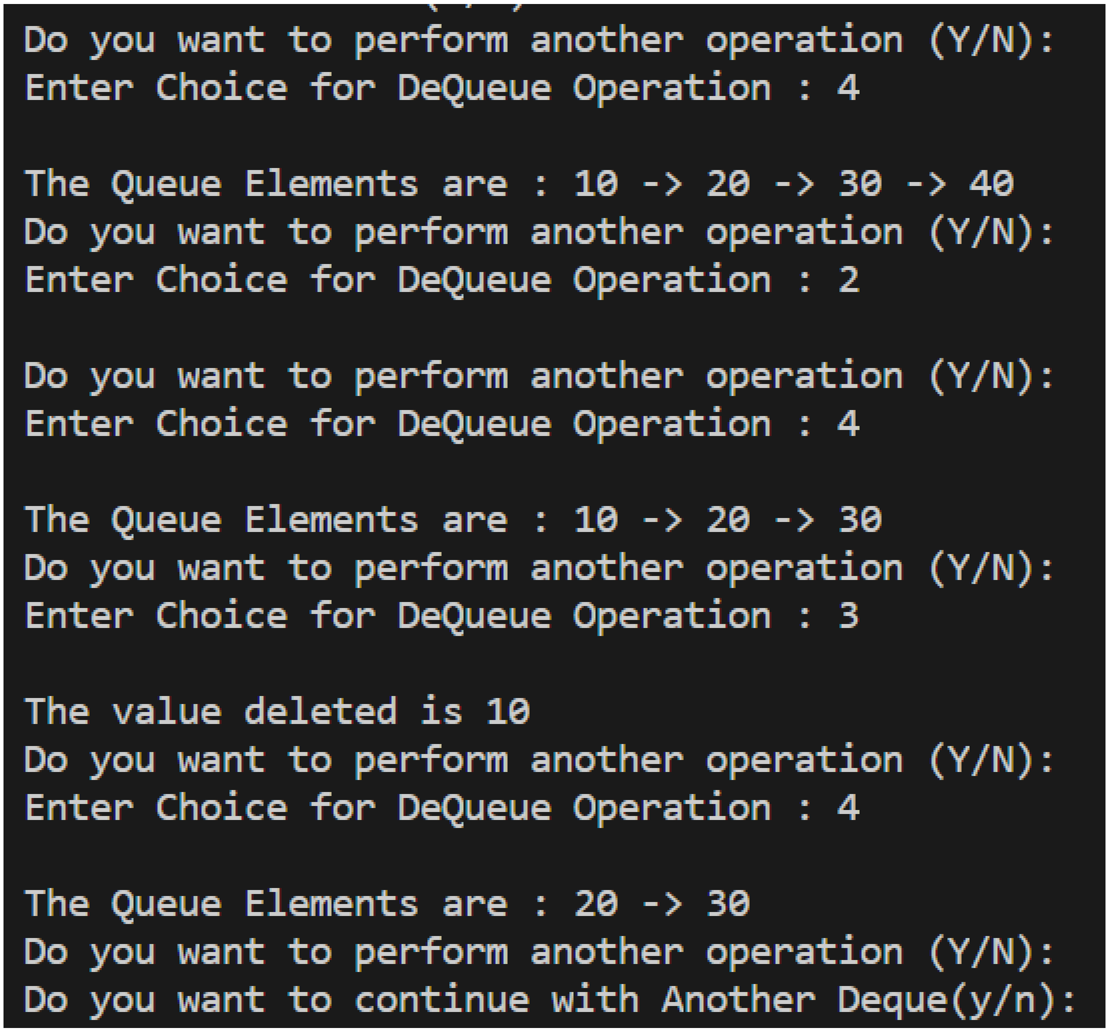


```c
        scanf("%d", &value);
        rear--;
        queue[rear] = value;
        printf("Continue Insertion(Y/N)?");
        ch = getch();
    } while (ch == 'y' || ch == 'Y');
}

int deQueueRear()
{
    int deleted;
    if (front == rear)
    {
        printf("\nQueue is Empty! Deletion Not Possible!");
        return 0;
    }
    front--;
    deleted = queue[front + 1];
    return deleted;
}

int deQueueFront()
{
    int deleted;
    if (front == rear)
    {
        printf("\nQueue is Empty! Deletion Not Possible!");
        return 0;
    }
    rear++;
    deleted = queue[rear - 1];
    return deleted;
}

void display()
{
    int i;
    if (front == rear)
        printf("\nQueue is Empty!");
    else
    {
        printf("\nThe Queue Elements are : ");
        for (i = rear; i < front - 1; i++)
        {
            printf("%d -> ", queue[i]);
        }
        printf("%d", queue[i]);
    }
}
```

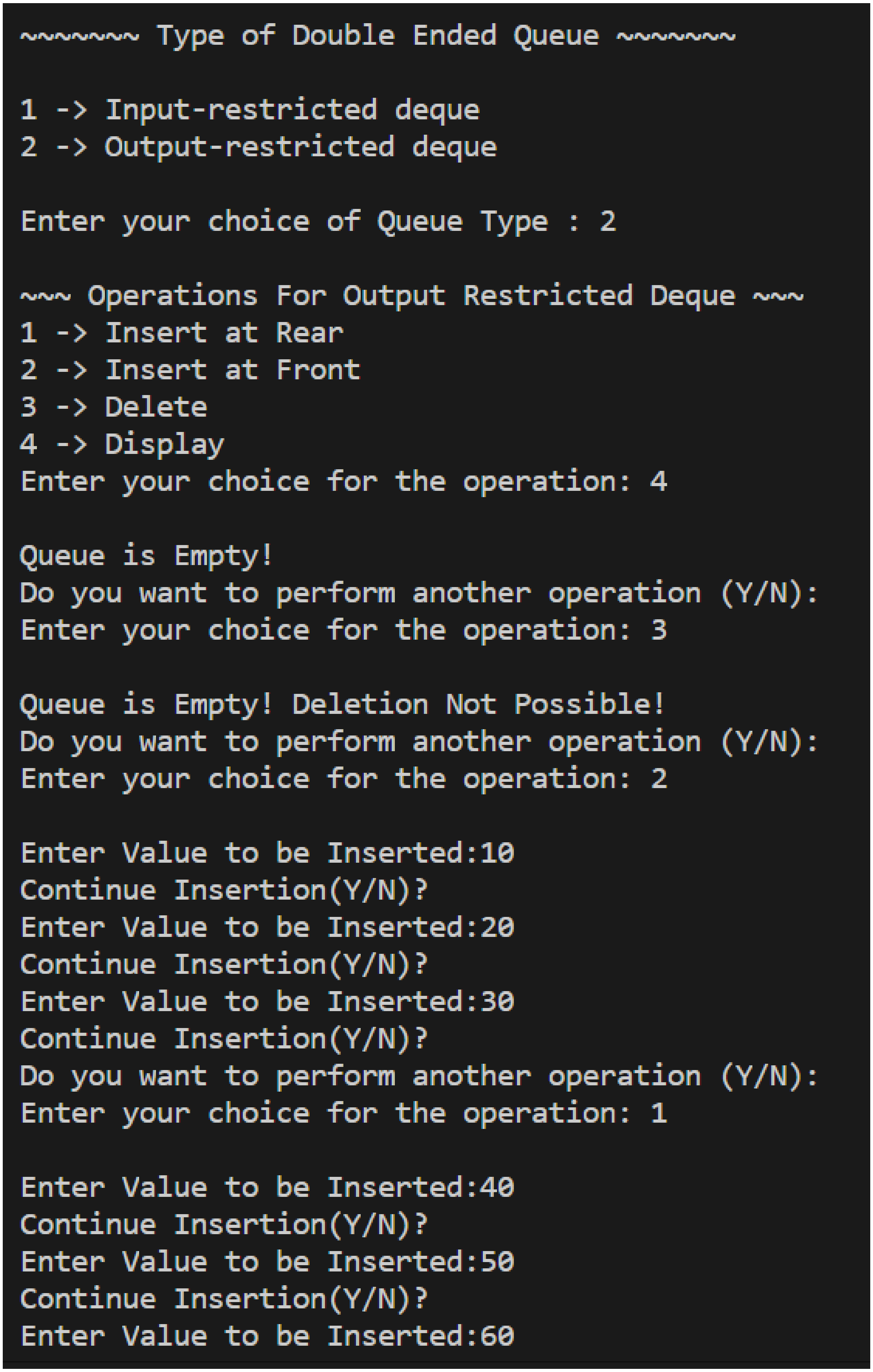
Example Test Cases:



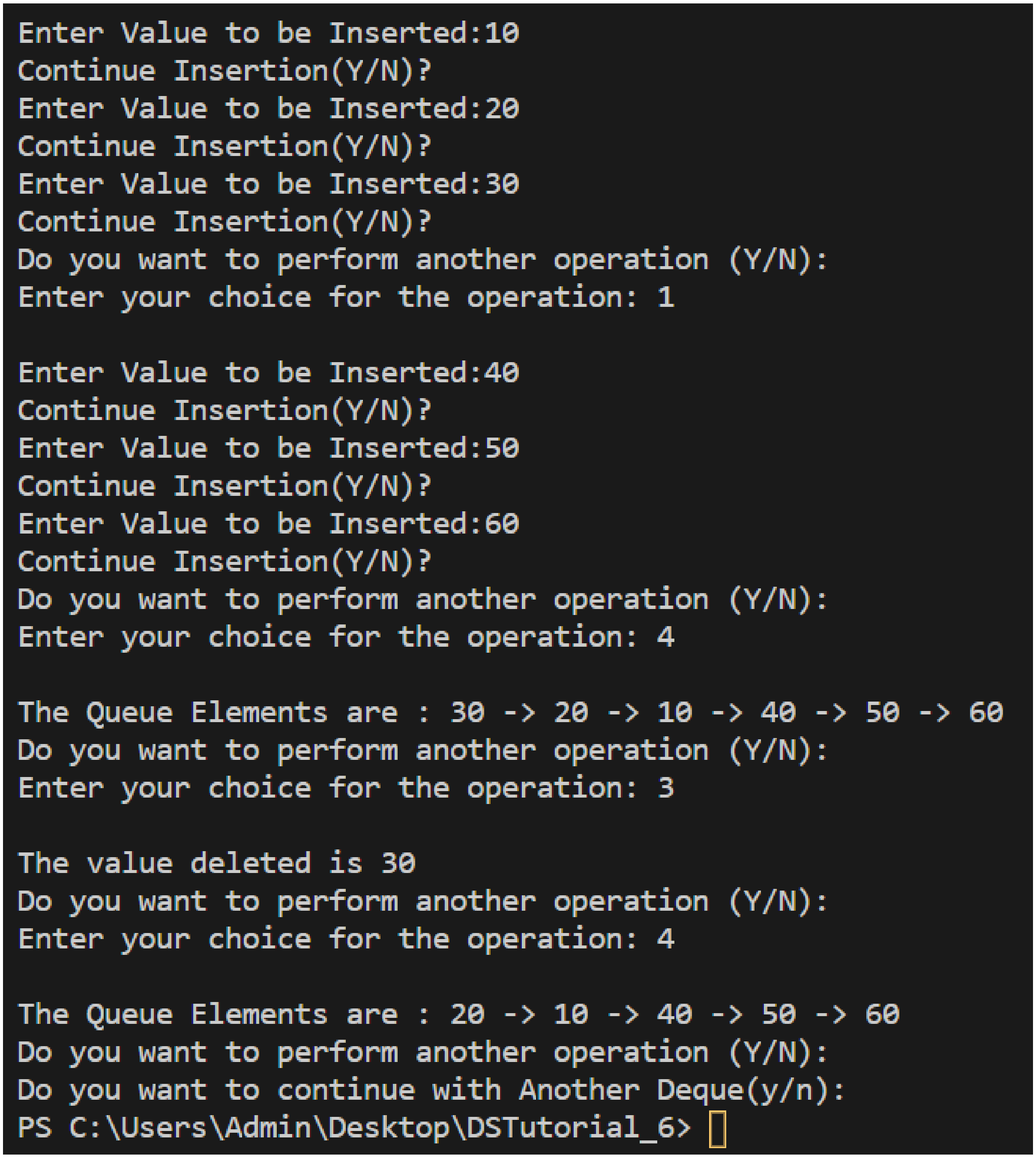
```
~~~~~~~ Type of Double Ended Queue ~~~~~~~

1 -> Input-restricted deque
2 -> Output-restricted deque

Enter your choice of Queue Type : 1

~~~ Operations For Input Restricted Deque ~~~
1 -> Insert
2 -> Delete from Rear
3 -> Delete from Front
4 ->  Display
Enter Choice for DeQueue Operation : 4

Queue is Empty!
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 2

Queue is Empty! Deletion Not Possible!
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 1

Enter Value to be Inserted:10
Continue Insertion(Y/N)?
Enter Value to be Inserted:20
Continue Insertion(Y/N)?
Enter Value to be Inserted:30
Continue Insertion(Y/N)?
Enter Value to be Inserted:40
Continue Insertion(Y/N)?
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 4

The Queue Elements are : 10 -> 20 -> 30 -> 40
```



```
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 4

The Queue Elements are : 10 -> 20 -> 30 -> 40
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 2

Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 4

The Queue Elements are : 10 -> 20 -> 30
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 3

The value deleted is 10
Do you want to perform another operation (Y/N):
Enter Choice for DeQueue Operation : 4

The Queue Elements are : 20 -> 30
Do you want to perform another operation (Y/N):
Do you want to continue with Another Deque(y/n):
```



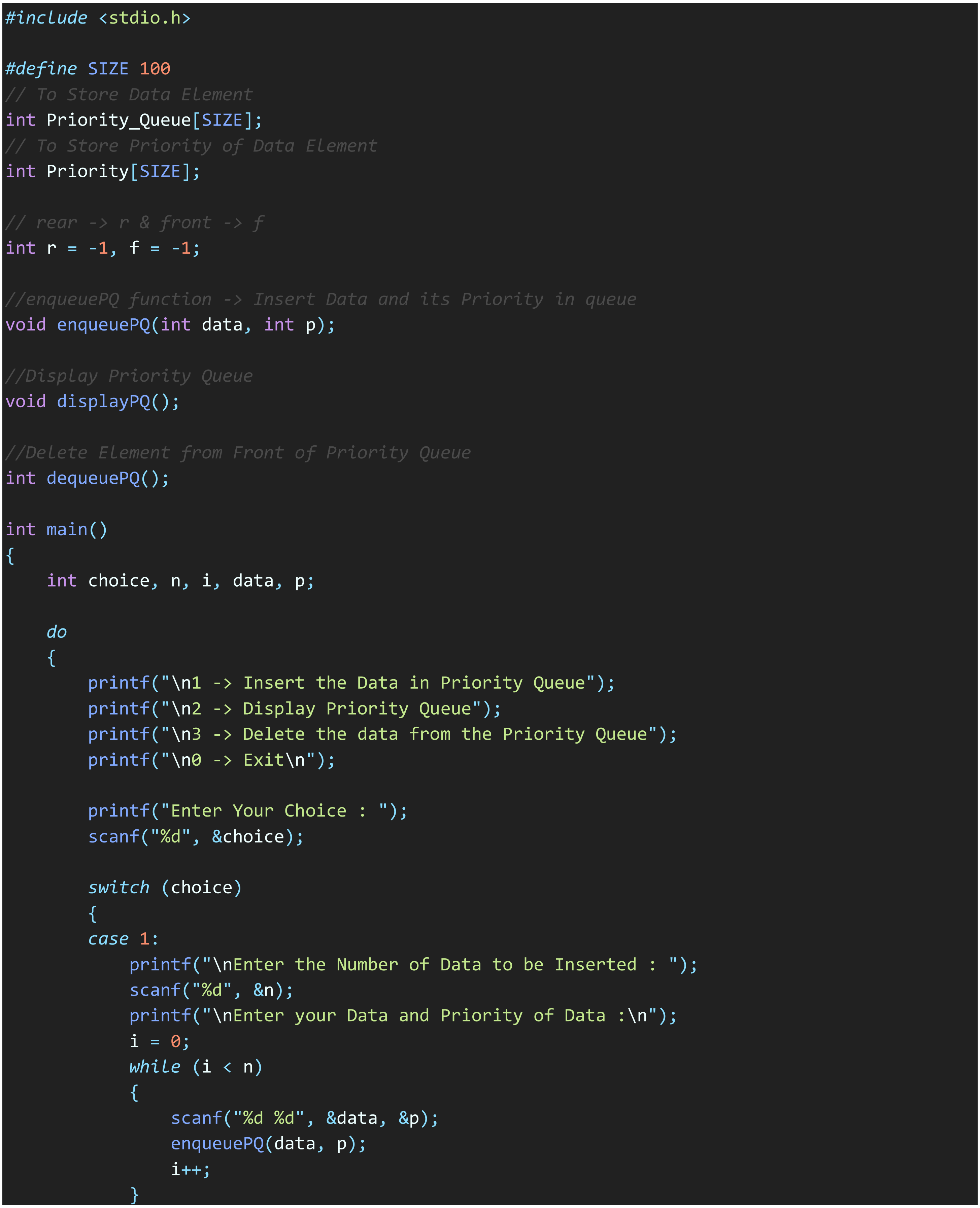
```
~~~~~~~ Type of Double Ended Queue ~~~~~~~

1 -> Input-restricted deque
2 -> Output-restricted deque

Enter your choice of Queue Type : 2

~~~ Operations For Output Restricted Deque ~~~
1 -> Insert at Rear
2 -> Insert at Front
3 -> Delete
4 -> Display
Enter your choice for the operation: 4

Queue is Empty!
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 3

Queue is Empty! Deletion Not Possible!
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 2

Enter Value to be Inserted:10
Continue Insertion(Y/N)?
Enter Value to be Inserted:20
Continue Insertion(Y/N)?
Enter Value to be Inserted:30
Continue Insertion(Y/N)?
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 1

Enter Value to be Inserted:40
Continue Insertion(Y/N)?
Enter Value to be Inserted:50
Continue Insertion(Y/N)?
Enter Value to be Inserted:60
```



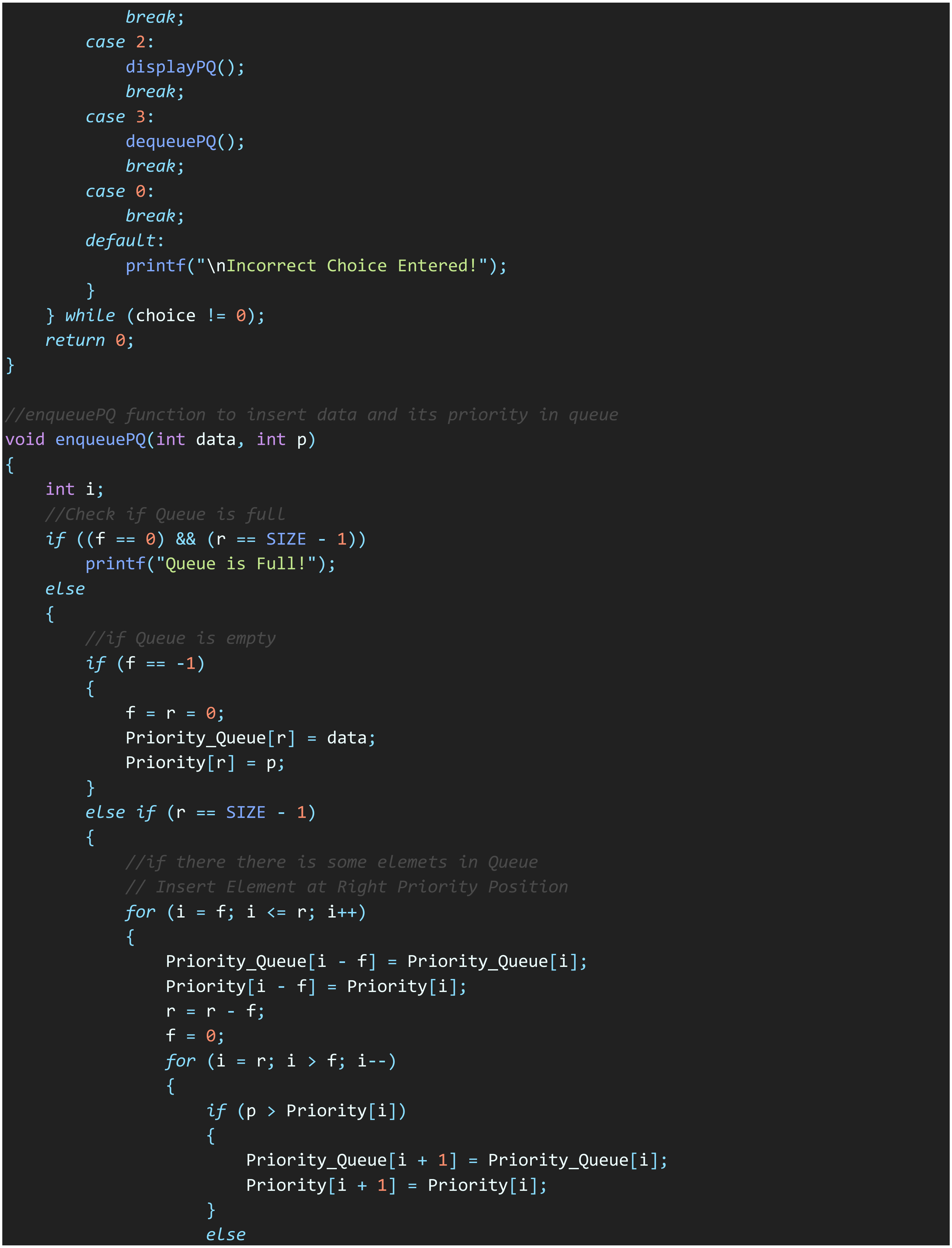
```
Enter Value to be Inserted:10
Continue Insertion(Y/N)?
Enter Value to be Inserted:20
Continue Insertion(Y/N)?
Enter Value to be Inserted:30
Continue Insertion(Y/N)?
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 1

Enter Value to be Inserted:40
Continue Insertion(Y/N)?
Enter Value to be Inserted:50
Continue Insertion(Y/N)?
Enter Value to be Inserted:60
Continue Insertion(Y/N)?
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 4

The Queue Elements are : 30 -> 20 -> 10 -> 40 -> 50 -> 60
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 3

The value deleted is 30
Do you want to perform another operation (Y/N):
Enter your choice for the operation: 4

The Queue Elements are : 20 -> 10 -> 40 -> 50 -> 60
Do you want to perform another operation (Y/N):
Do you want to continue with Another Deque(y/n):
PS C:\Users\Admin\Desktop\DSTutorial_6>
```

3. Perform above operations to implement Priority Queue

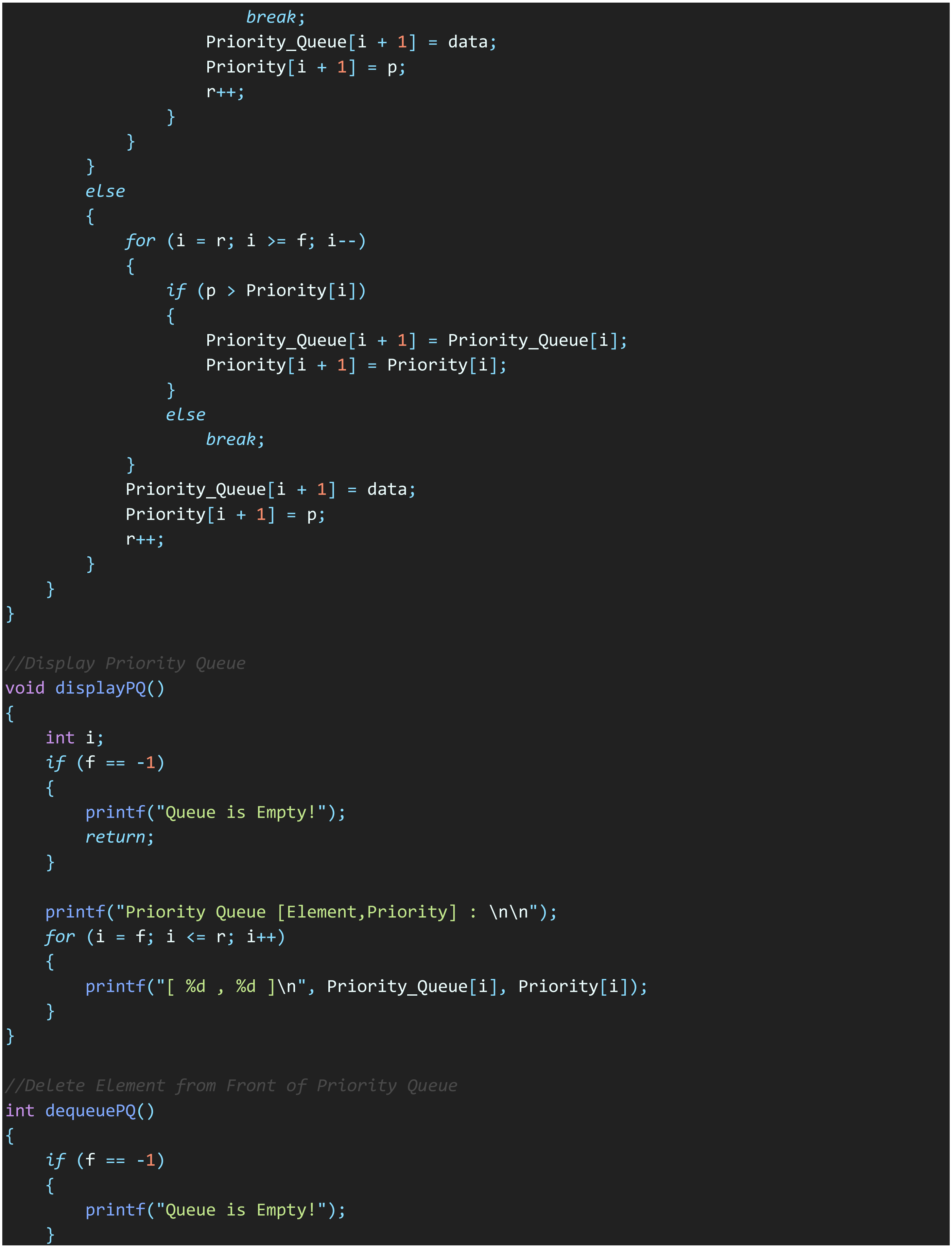
Code:



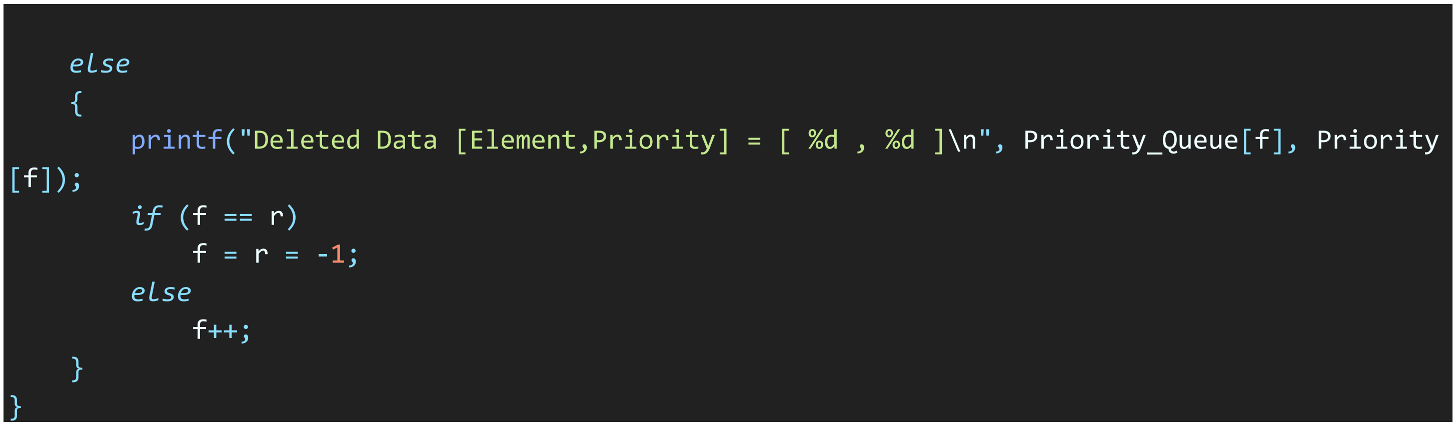
```c
#include <stdio.h>

#define SIZE 100
// To Store Data Element
int Priority_Queue[SIZE];
// To Store Priority of Data Element
int Priority[SIZE];

// rear -> r & front -> f
int r = -1, f = -1;

//enqueuePQ function -> Insert Data and its Priority in queue
void enqueuePQ(int data, int p);

//Display Priority Queue
void displayPQ();

//Delete Element from Front of Priority Queue
int dequeuePQ();

int main()
{
    int choice, n, i, data, p;

    do
    {
        printf("\n1 -> Insert the Data in Priority Queue");
        printf("\n2 -> Display Priority Queue");
        printf("\n3 -> Delete the data from the Priority Queue");
        printf("\n0 -> Exit\n");

        printf("Enter Your Choice : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("\nEnter the Number of Data to be Inserted : ");
            scanf("%d", &n);
            printf("\nEnter your Data and Priority of Data :\n");
            i = 0;
            while (i < n)
            {
                scanf("%d %d", &data, &p);
                enqueuePQ(data, p);
                i++;
            }
```



```
            break;
        case 2:
            displayPQ();
            break;
        case 3:
            dequeuePQ();
            break;
        case 0:
            break;
        default:
            printf("\nIncorrect Choice Entered!");
        }
    } while (choice != 0);
    return 0;
}

//enqueuePQ function to insert data and its priority in queue
void enqueuePQ(int data, int p)
{
    int i;
    //Check if Queue is full
    if ((f == 0) && (r == SIZE - 1))
        printf("Queue is Full!");
    else
    {
        //if Queue is empty
        if (f == -1)
        {
            f = r = 0;
            Priority_Queue[r] = data;
            Priority[r] = p;
        }
        else if (r == SIZE - 1)
        {
            //if there there is some elemets in Queue
            // Insert Element at Right Priority Position
            for (i = f; i <= r; i++)
            {
                Priority_Queue[i - f] = Priority_Queue[i];
                Priority[i - f] = Priority[i];
                r = r - f;
                f = 0;
                for (i = r; i > f; i--)
                {
                    if (p > Priority[i])
                    {
                        Priority_Queue[i + 1] = Priority_Queue[i];
                        Priority[i + 1] = Priority[i];
                    }
                    else
```

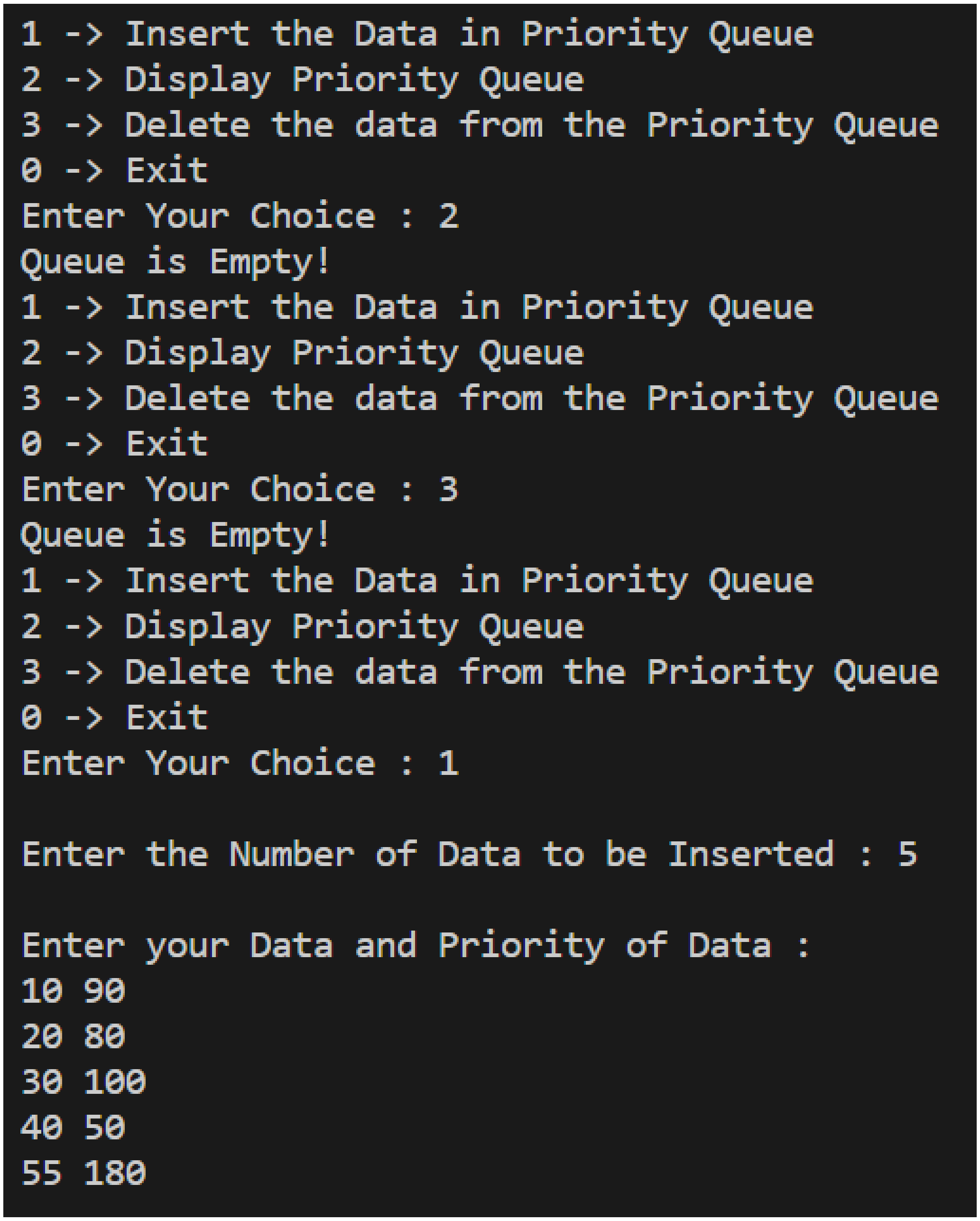


```
                        break;
                    Priority_Queue[i + 1] = data;
                    Priority[i + 1] = p;
                    r++;
                }
            }
        }
        else
        {
            for (i = r; i >= f; i--)
            {
                if (p > Priority[i])
                {
                    Priority_Queue[i + 1] = Priority_Queue[i];
                    Priority[i + 1] = Priority[i];
                }
                else
                    break;
            }
            Priority_Queue[i + 1] = data;
            Priority[i + 1] = p;
            r++;
        }
    }
}

//Display Priority Queue
void displayPQ()
{
    int i;
    if (f == -1)
    {
        printf("Queue is Empty!");
        return;
    }

    printf("Priority Queue [Element,Priority] : \n\n");
    for (i = f; i <= r; i++)
    {
        printf("[ %d , %d ]\n", Priority_Queue[i], Priority[i]);
    }
}

//Delete Element from Front of Priority Queue
int dequeuePQ()
{
    if (f == -1)
    {
        printf("Queue is Empty!");
    }
```

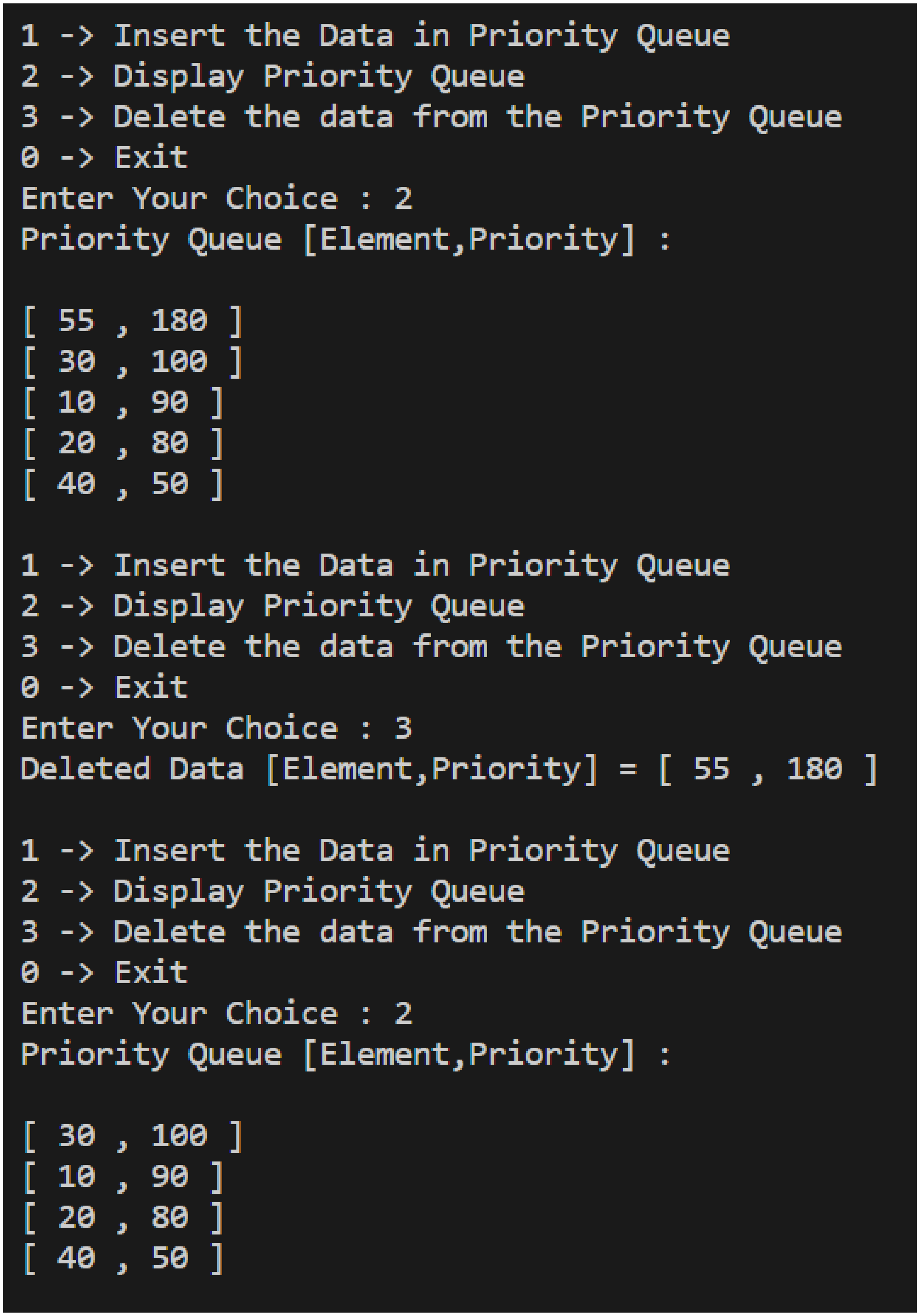


```
    else
    {
        printf("Deleted Data [Element,Priority] = [ %d , %d ]\n", Priority_Queue[f], Priority
[f]);
        if (f == r)
            f = r = -1;
        else
            f++;
    }
}
```

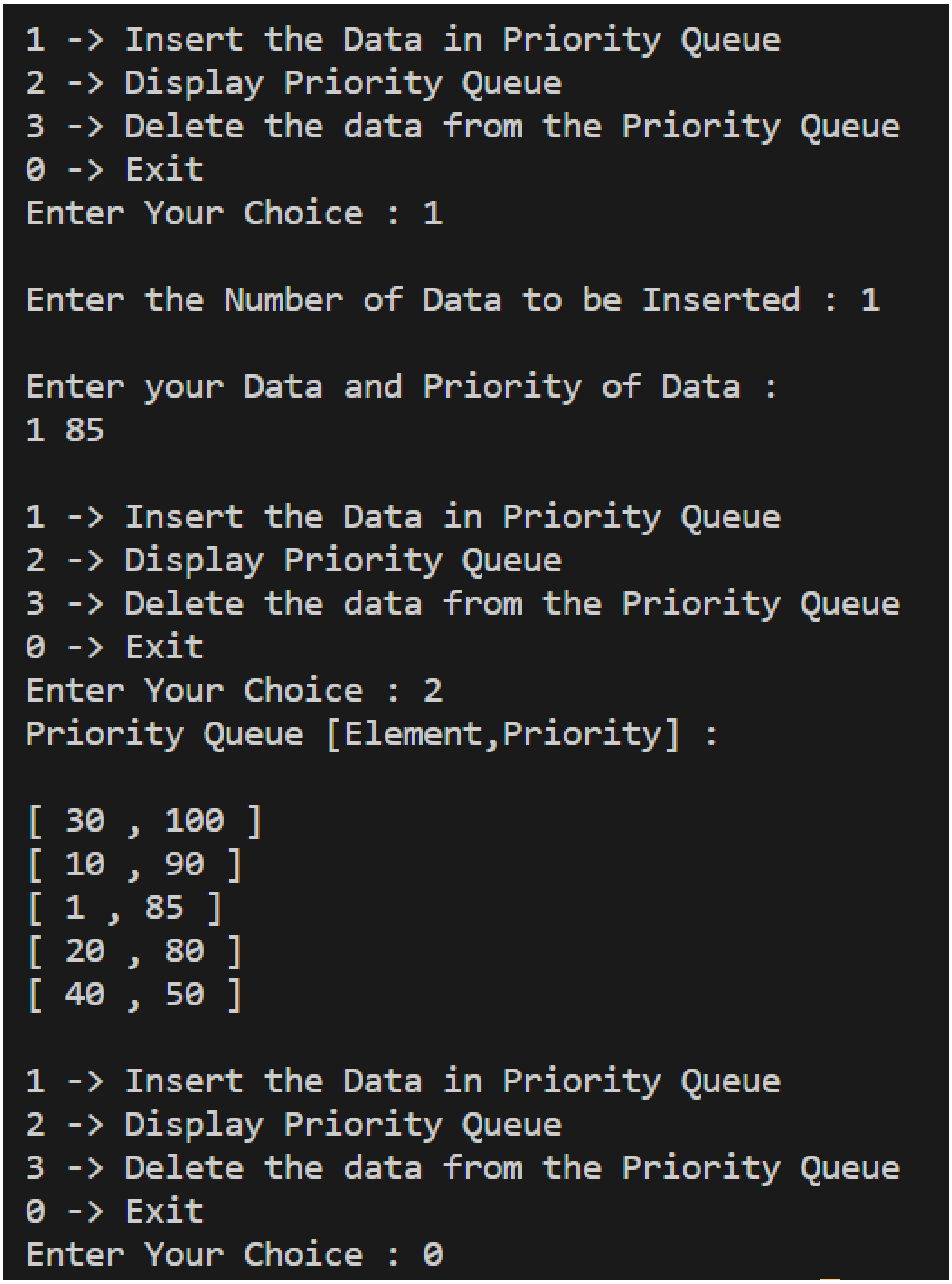
Example Test Cases:



```
1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 2
Queue is Empty!
1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 3
Queue is Empty!
1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 1

Enter the Number of Data to be Inserted : 5

Enter your Data and Priority of Data :
10 90
20 80
30 100
40 50
55 180
```



```
1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 2
Priority Queue [Element,Priority] :

[ 55 , 180 ]
[ 30 , 100 ]
[ 10 , 90 ]
[ 20 , 80 ]
[ 40 , 50 ]

1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 3
Deleted Data [Element,Priority] = [ 55 , 180 ]

1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 2
Priority Queue [Element,Priority] :

[ 30 , 100 ]
[ 10 , 90 ]
[ 20 , 80 ]
[ 40 , 50 ]
```



```
1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 1

Enter the Number of Data to be Inserted : 1

Enter your Data and Priority of Data :
1 85

1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 2
Priority Queue [Element,Priority] :

[ 30 , 100 ]
[ 10 , 90 ]
[ 1 , 85 ]
[ 20 , 80 ]
[ 40 , 50 ]

1 -> Insert the Data in Priority Queue
2 -> Display Priority Queue
3 -> Delete the data from the Priority Queue
0 -> Exit
Enter Your Choice : 0
```

*We have **successfully Implemented** and **Verified** Queue, Deque and Priority Queue in C.*

Submitted By:
Roll Number: **U19CS012** (*D-12*)
Name: *Bhagya Rana*