PRN: 2020BTEIT00041

Queue Assignment:

1.Write a program to implement queue using array.

CODE:

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

/*
    Queue using Array implementation
*/

#include <iostream>

using namespace std;

#define n 100

class queue
{
public:
    int bottom;
    int *arr;
    queue()
    {
        arr = new int[n];
        bottom = 0;
    }

    void enqueue(int data)
    {
        if (bottom == n)
        {
            cout << "Queue is full" << endl;
            return;
        }
        else
        {
            arr[bottom] = data;
            bottom++;
        }
        return;
    }

    void dequeue()
    {
        if (bottom == 0)
        {
            cout << "Queue is empty" << endl;
            return;
        }
```

```
        else
        {
            for (int i = 0; i < bottom - 1; i++)
            {
                arr[i] = arr[i + 1];
            }
            bottom--;
        }
        return;
    }

    void displayqueue()
    {
        for (int i = 0; i < bottom; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
        return;
    }
};

int main()
{
    queue q1;

    int a = 0;

    while (a != 4)
    {
        cout << "----------------------Menu----------------------\n";
        cout << "1. Enqueue" << endl
             << "2. Dequeue" << endl
             << "3. Display" << endl
             << "4. Exit" << endl;
        cin >> a;
        switch (a)
        {
        case 1:
            int data;
            cout << "Enter data : " << endl;
            cin >> data;
            q1.enqueue(data);
            break;

        case 2:
            q1.dequeue();
            break;
```

```
 97
 98          case 3:
 99              q1.displayqueue();
100              break;
101
102          case 4:
103              cout << "exited" << endl;
104              break;
105
106          default:
107              cout << "Enter Valid choice" << endl;
108          }
109      }
110
111      return 0;
112  }
```

OUTPUT:

```
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
1
Enter data :
2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
1
Enter data :
3
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
1
Enter data :
4
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
3
2 3 4
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
3
3 4
```

2.Write a program to implement Circular Queue using array.
CODE:

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

/*
    Circular Queue  -  Implementation of a circular queue using an Array
*/

#include <iostream>

using namespace std;

#define n 100

class queue
{
public:
    int bottom;
    int *arr;
    queue()
    {
        arr = new int[n];
        bottom = 0;
    }

    void enqueue(int data)
    {
        if (bottom == n)
        {
            cout << "Queue is full" << endl;
            return;
        }
        else
        {
            arr[bottom] = data;
            bottom++;
        }
        return;
    }

    void dequeue()
    {
        if (bottom == 0)
        {
            cout << "Queue is empty" << endl;
            return;
        }
        else
```

```
        {
            for (int i = 0; i < bottom - 1; i++)
            {
                arr[i] = arr[i + 1];
            }
            bottom--;
        }
        return;
    }

    void displayqueue()
    {
        for (int i = 0; i < bottom; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
        return;
    }
};

int main()
{
    queue q1;

    int a = 0;

    while (a != 4)
    {
        cout << "--------------------Menu--------------------\n";
        cout << "1. Enqueue" << endl
             << "2. Dequeue" << endl
             << "3. Display" << endl
             << "4. Exit" << endl;
        cin >> a;
        switch (a)
        {
        case 1:
            int data;
            cout << "Enter data : " << endl;
            cin >> data;
            q1.enqueue(data);
            break;

        case 2:
            q1.dequeue();
            break;
```

```
97
98          case 3:
99              q1.displayqueue();
100             break;
101
102         case 4:
103             cout << "exited" << endl;
104             break;
105
106         default:
107             cout << "Enter Valid choice" << endl;
108         }
109     }
110
111     return 0;
112 }
```

OUTPUT:

```
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
1
Enter data :
1
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
1
Enter data :
2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
1
Enter data :
2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
3
1 2 2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
3
2 2
```

3.Write a program to implement priority Queue (Ascending and Descending)

CODE:

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

/*
    Priority Queue - Implementation using linkedlist
*/

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

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

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

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

// Priority Queue class
class PriorityQueue{
public:
    Node* head;

    PriorityQueue(){
        this->head = NULL;
    }

    void enqueue(int data, int priority);
    int dequeue();
    void Display();
};

// enqueue function
void PriorityQueue::enqueue(int data, int priority){
    // create a new node
```

```
49      Node *newNode = new Node(data, priority);
50
51      // if queue is empty
52      if(this->head == NULL){
53          this->head = newNode;
54          return;
55      }
56
57      // if queue is not empty
58      Node *temp = this->head;
59      Node *prev = NULL;
60
61      // traverse the queue to find the right position to insert the new node in the queue
62      while(temp != NULL){
63          if(temp->priority > priority){
64              break;
65          }
66          prev = temp;
67          temp = temp->next;
68      }
69
70      // if new node's priority is less than head's priority
71      if(prev == NULL){
72          newNode->next = this->head;
73          this->head = newNode;
74      }
75      // if new node's priority is greater than head's priority
76      else{
77          prev->next = newNode;
78          newNode->next = temp;
79      }
80
81      return;
82  }
83
84  int PriorityQueue::dequeue(){
85      // if queue is empty
86      if(this->head == NULL){
87          cout << "Queue is empty" << endl;
88          return -1;
89      }
90
91      // if queue is not empty
92      // store the head node in a temp variable
93      Node *temp = this->head;
94
95      // store the head node's data in a temp variable
```

```
96      int data = temp->data;
97
98      // make the front as the next of the front
99      this->head = this->head->next;
100
101     // delete the temp node
102     delete temp;
103
104     return data;
105 }
106
107 void PriorityQueue::Display(){
108     // if queue is empty
109     if(this->head == NULL){
110         cout << "Queue is empty" << endl;
111         return;
112     }
113
114     // if queue is not empty
115     Node *temp = this->head;
116
117     // traverse the queue and print the data of each node
118     while(temp != NULL){
119         cout << temp->data << " ";
120         temp = temp->next;
121     }
122     cout << endl;
123 }
124
125 int main(){
126     PriorityQueue pq;
127
128     // Menu driven program to implement a priority queue
129
130     int choice, data, priority;
131
132     do{
133         cout<<"--------------------Menu--------------------\n";
134         cout << "1. Enqueue" << endl;
135         cout << "2. Dequeue" << endl;
136         cout << "3. Display" << endl;
137         cout << "4. Exit" << endl;
138
139         cout << "Enter your choice: ";
140         cin >> choice;
141
142         switch(choice){
143             case 1:
144                 cout << "Enter data: ";
```

```
                cin >> data;
                cout << "Enter priority: ";
                cin >> priority;
                pq.enqueue(data, priority);
                break;
            case 2:
                cout<<"Dequeued element: "<<pq.dequeue()<<endl;
                break;
            case 3:
                pq.Display();
                break;
            case 4:
                break;
            default:
                cout << "Invalid choice" << endl;
        }
    }while(choice != 4);

    return 0;
}
```

OUTPUT:

```
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter data: 1
Enter priority: 3
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter data: 2
Enter priority: 2
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter data: 3
Enter priority: 1
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
3 2 1
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued element: 3
--------------------Menu--------------------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
2 1
```

4. Write a program to implement Deque

CODE:

```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();
```

```
};

// 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;
```

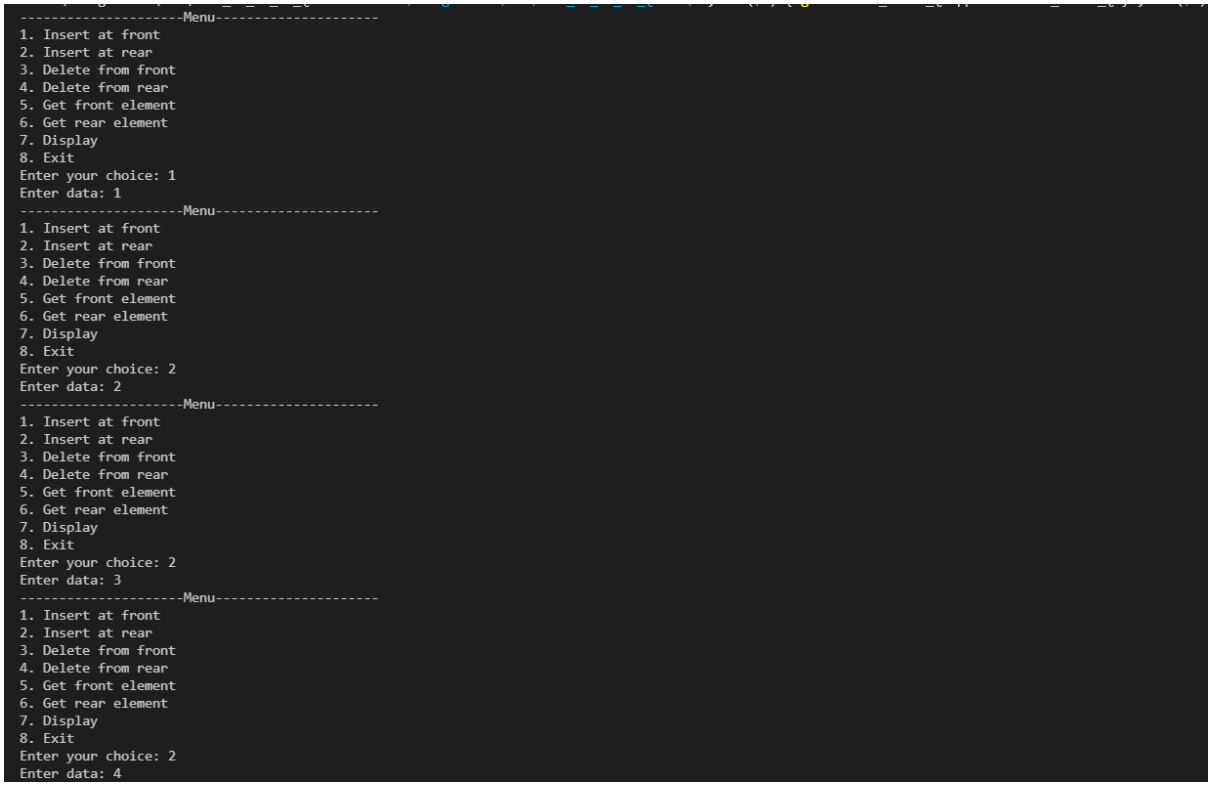
```
        return;
    }
    // If Deque has only one element
    else if(this->front == this->rear){
        delete this->front;
        this->front = NULL;
        this->rear = NULL;
    }
    // If Deque has more than one element
    else{
        // Store pointer to old front
        Node *old_front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

```
142
143         // Move rear to point to last element
144         this->rear = last;
145
146         // Change next of last element to NULL
147         this->rear->next = NULL;
148     }
149 }
150
151 // Returns element from front of Deque
152 int Deque::getFront(){
153     // If Deque is empty
154     if(this->front == NULL){
155         cout << "Deque is empty" << endl;
156         return -1;
157     }
158     // If Deque has only one element
159     else if(this->front == this->rear){
160         return this->front->data;
161     }
162     // If Deque has more than one element
163     else{
164         return this->front->data;
165     }
166 }
167
168 // Returns element from rear of Deque
169 int Deque::getRear(){
170     // If Deque is empty
171     if(this->front == NULL){
172         cout << "Deque is empty" << endl;
173         return -1;
174     }
175     // If Deque has only one element
176     else if(this->front == this->rear){
177         return this->front->data;
178     }
179     // If Deque has more than one element
180     else{
181         return this->rear->data;
182     }
183 }
184
185 // Displays Deque
186 void Deque::Display(){
187     // If Deque is empty
188     if(this->front == NULL){
```

```
189         cout << "Deque is empty" << endl;
190         return;
191     }
192     // If Deque has only one element
193     else if(this->front == this->rear){
194         cout << this->front->data << endl;
195     }
196     // If Deque has more than one element
197     else{
198         // Store pointer to last element
199         Node* first = this->front;
200         Node *last = this->front;
201 
202         // Find second last element
203         while(last->next != this->rear){
204             last = last->next;
205         }
206 
207         // Display elements from front to second last
208         while(first != last){
209             cout << first->data << " ";
210             first = first->next;
211         }
212 
213         // Display last element
214         cout << this->rear->data << endl;
215 
216         // Move front and rear to point to NULL
217         first = NULL;
218         last = NULL;
219     }
220 }
221 
222 int main(){
223     Deque d;
224 
225     // Menu driven program
226     int choice;
227 
228     do{
229         cout<<"--------------------Menu--------------------\n";
230         cout << "1. Insert at front" << endl;
231         cout << "2. Insert at rear" << endl;
232         cout << "3. Delete from front" << endl;
233         cout << "4. Delete from rear" << endl;
234         cout << "5. Get front element" << endl;
```

```
235         cout << "6. Get rear element" << endl;
236         cout << "7. Display" << endl;
237         cout << "8. Exit" << endl;
238 
239         cout << "Enter your choice: ";
240         cin >> choice;
241 
242         switch(choice){
243             case 1:
244                 int dataFront;
245                 cout << "Enter data: ";
246                 cin >> dataFront;
247                 d.insertFront(dataFront);
248                 break;
249             case 2:
250                 int dataRear;
251                 cout << "Enter data: ";
252                 cin >> dataRear;
253                 d.insertRear(dataRear);
254                 break;
255             case 3:
256                 d.deleteFront();
257                 break;
258             case 4:
259                 d.deleteRear();
260                 break;
261             case 5:
262                 cout << "Front element is: " << d.getFront() << endl;
263                 break;
264             case 6:
265                 cout << "Rear element is: " << d.getRear() << endl;
266                 break;
267             case 7:
268                 d.Display();
269             case 8:
270                 cout<<"Exiting..."<<endl;
271                 break;
272             default:
273                 cout << "Wrong choice" << endl;
274         }
275     }while(choice != 8);
276 
277     return 0;
278 }
```

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

5. Write a program to implement queue using linked list or queADT

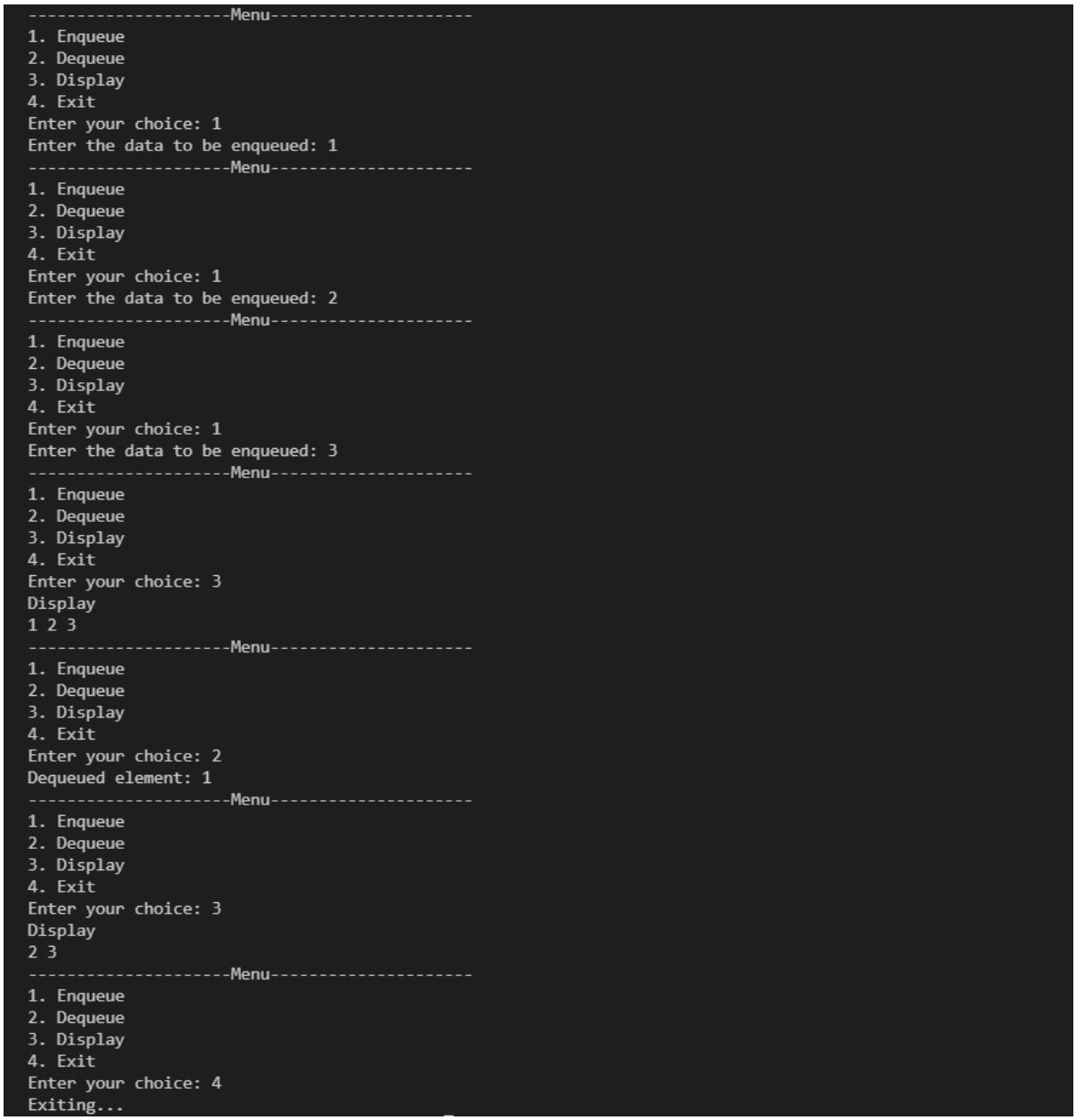
CODE:

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

/*
    Queue using Linked List implementation
*/

#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;
    }
};

// Queue class
class Queue{
public:
    Node* front;
    Node* rear;

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

    void enqueue(int data);
    int dequeue();
    void display();
};

// Function to enqueue an element in the queue using LL implementation of Queue
void Queue::enqueue(int data){
    // Create a new node
    Node* new_node = new Node(data);
```

```
50
51      // If queue is empty
52      if(this->front == NULL){
53          // Make the new node as the front and rear
54          this->front = new_node;
55          this->rear = new_node;
56      }
57      else{
58          // Make the new node as the rear
59          this->rear->next = new_node;
60          this->rear = new_node;
61      }
62  }
63
64  // Function to dequeue an element from the queue using LL implementation of Queue
65  int Queue::dequeue(){
66      // if queue is empty
67      if(this->front == NULL){
68          cout<<"Queue is empty"<<endl;
69          return -1;
70      }
71      else{
72          // Store the data of the front node
73          int data = this->front->data;
74
75          // Make the next node as the front
76          this->front = this->front->next;
77
78          // If queue is empty
79          if(this->front == NULL){
80              this->rear = NULL;
81          }
82
83          return data;
84      }
85  }
86
87  // Display the Queue
88  void Queue::display(){
89      // If queue is empty
90      if(this->front == NULL){
91          cout<<"Queue is empty"<<endl;
92          return;
93      }
94
95      // Create a temporary node
96      Node* temp = this->front;
```

```
97
98      // Traverse the queue
99      while(temp != NULL){
100         cout<<temp->data<<" ";
101         temp = temp->next;
102     }
103     cout<<endl;
104 }
105
106 int main(){
107     Queue q;
108
109     // Menu driven program to perform operations on the queue using LL implementation of Q
110     int choice;
111     do{
112         cout<<"--------------------Menu--------------------\n";
113         cout<<"1. Enqueue"<<endl;
114         cout<<"2. Dequeue"<<endl;
115         cout<<"3. Display"<<endl;
116         cout<<"4. Exit"<<endl;
117
118
119         cout<<"Enter your choice: ";
120         cin>>choice;
121
122         switch(choice){
123             case 1:
124                 int data;
125                 cout<<"Enter the data to be enqueued: ";
126                 cin>>data;
127                 q.enqueue(data);
128                 break;
129             case 2:
130                 cout<<"Dequeued element: "<<q.dequeue()<<endl;
131                 break;
132             case 3:
133                 cout<<"Display"<<endl;
134                 q.display();
135                 break;
136             case 4:
137                 cout<<"Exiting..."<<endl;
138                 break;
139             default:
140                 cout<<"Invalid choice"<<endl;
141         }
142     }while(choice != 4);
```

OUTPUT:


----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the data to be enqueued: 1
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the data to be enqueued: 2
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the data to be enqueued: 3
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Display
1 2 3
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued element: 1
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Display
2 3
----------Menu----------
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
Exiting...