MUMBAI EDUCATIONAL TRUST

THE MET LEAGUE OF COLLEGES
MET
AS SHARP AS YOU CAN GET
Bhujbal Knowledge City

MET Institute of Computer Science

Name	Shubham Sarang
Roll No	1345
Topic	Queues
Programs	1) Ordinary Queues 2) Circular Queues 3) Double-Ended Queues 4) Priority Queues

Ordinary Queue:

```
#include<iostream>

#include<conio.h>

#define MAX 5


using namespace std;


// Queue template


class OQueue

{

	int front, rear, a[MAX];

	public:

		OQueue()

		{

			front = -1;

			rear = -1;

		}
```

```
        void Enqueue(int x);
        void Dequeue();
        void PeekFront();
        void PeekRear();
        void Display();
        int Full();
        int Empty();
};

//Functions

void OQueue::Enqueue(int x)
{
    if(Full())
    {
        cout<<"Queue Overflow!!";
        return;
    }
    if(front == -1)
    {
        front++;
    }
```

```
        rear++;
        a[rear]=x;
}

void OQueue::Dequeue()
{
        if(Empty())
        {
                cout<<"Queue underflow!!";
                return;
        }
        int tmp = a[front];

        if(front == rear)
        {
                front = -1;
                rear = -1;
        }
        else
        {
                front++;
        }
```

```
        cout<<"Element "<<tmp<<" dequeued";
}

int OQueue::Full()
{
        int result = rear == MAX-1 ? 1:0;
        return result;
}

int OQueue::Empty()
{
        int result = rear == -1 ? 1:0;
        return result;
}

void OQueue::PeekFront()
{
        if(Empty())
        {
                cout<<"Queue underflow!!";
                return;
        }
```

```
        cout<<"Element at the front: "<<a[front];
}

void OQueue::PeekRear()
{
        if(Empty())
        {
                cout<<"Queue underflow!!";
                return;
        }
        cout<<"Element at the rear: "<<a[rear];
}

void OQueue::Display()
{
        if(Empty())
        {
                cout<<"Queue underflow!!";
                return;
        }

        cout<<"Queue: \n";
```

```
        for(int i=front; i<=rear; i++)
        {
                cout<<a[i]<<"\n";
        }
}

//Menu

int main()
{
        OQueue o;
        int num, ch;
        while(1)
        {
                system("cls");
                cout<<"***Ordinary Queue***\n\n";
                cout<<"1. Perfrom Enqueue on queue\n";
                cout<<"2. Perfrom Dequeue on queue\n";
                cout<<"3. PeekFront of queue\n";
                cout<<"4. PeekRear of queue\n";
                cout<<"5. Display queue\n";
                cout<<"6. Exit\n\n";
```

```
cout<<"Enter the choice you've made: ";
cin>>ch;
switch(ch)
{
        case 1:
                cout<<"Add Element: ";
                cin>>num;
                o.Enqueue(num);
                getch();
                break;
        case 2:
                o.Dequeue();
                getch();
                break;
        case 3:
                o.PeekFront();
                getch();
                break;
        case 4:
                o.PeekRear();
                getch();
                break;
```

```
                case 5:
                        o.Display();
                        getch();
                        break;
                case 6:
                        exit(1);
                default:
                        cout<<"Wrong option bruv";
                        getch();
                        break;
            }
        }
}
```

Output:

MET Institute of Computer Science

Menu:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: _
```

Enqueue:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: 1
Add Element: 20_
```

Queue overflow:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: 1
Add Element: 30
Queue Overflow!!
```

Dequeue:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: 2
Element 20 dequeued
```

Queue:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: 2
Queue underflow!!
```

MET Institute of Computer Science

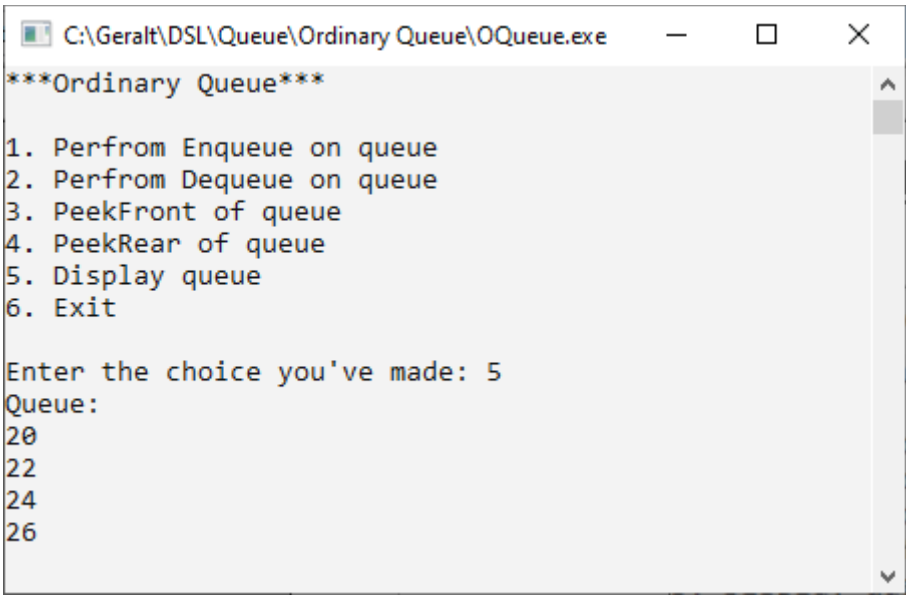
PeekFront:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: 3
Element at the front: 20
```

PeekRear:

```
C:\Geralt\DSL\Queue\Ordinary Queue\OQueue.exe
***Ordinary Queue***

1. Perfrom Enqueue on queue
2. Perfrom Dequeue on queue
3. PeekFront of queue
4. PeekRear of queue
5. Display queue
6. Exit

Enter the choice you've made: 4
Element at the rear: 26
```

Display:



Circular Queue:

Code:

```
#include <iostream>
#include <conio.h>
#define MAX 4
using namespace std;

// 2. Queues Template
class CQueue
{
    int A[MAX];
    int front;
    int rear;
```

```
    int cnt;


public:
    CQueue()
    {
        front = -1;
        rear = -1;
        cnt = 0;
    }
    void Enqueue(int x);
    void Dequeue();
    void PeekFront();
    void PeekRear();
    void Display();
    int Full();
    int Empty();
}; // 3. Functions
int CQueue ::Full()
{
    if (cnt == MAX)
    {
        return 1;
```

```
    }
    else
    {
        return 0;
    }
} // end of the Full
int CQueue ::Empty()
{
    if (cnt == 0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
} // end of the Empty
void CQueue ::Enqueue(int x)
{
    if (Full())
    {
        cout << "Queue OverFlow";
```

```
        return;
    }
    if (front == -1)
    {
        front++;
    }
    if (rear == MAX - 1)
    {
        rear = 0;
    }
    else
    {
        rear++;
    }
    A[rear] = x;
    cnt++;
} // End Of Enqueue
void CQueue ::Dequeue()
{
    if (Empty())
    {
        cout << "Underflow!!";
```

```
        return;
    }
    int tmp = A[front];
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else
    {
        if (front == MAX - 1)
        {
            front = 0;
        }
        else
        {
            front++;
        }
    }
    cout << "Dequeued Element is : " << tmp;
    cnt--;
} // end of Dequeue
```

```
void CQueue ::Display()
{
    if (Empty())
    {
        cout << "Empty!!";
        return;
    }
    int j = front;
    for (int i = 1; i <= cnt; i++)
    {
        cout << A[j] << " ";
        if (j == MAX - 1)
        {
            j = 0;
        }
        else
        {
            j++;
        }
    }
} // end of the Display
void CQueue ::PeekFront()
```

```
{
    if (Empty())
    {
        cout << "Underflow!!";
    }
    else
    {
        cout << "Element at the Front is " << A[front];
    }
} // end of the PeekFront
void CQueue ::PeekRear()
{
    if (Empty())
    {
        cout << "Underflow!!";
    }
    else
    {
        cout << "Element at the Rear is " << A[rear];
    }
} // end of the PeekRear
// 4. Menu
```

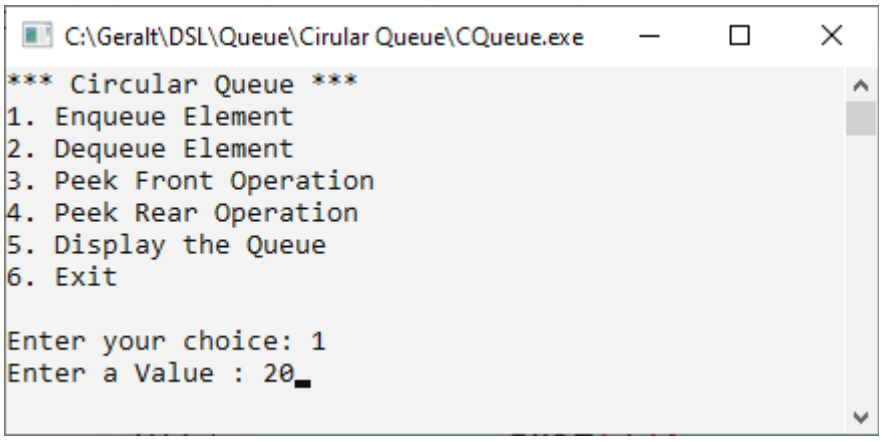
```
int main()
{
    CQueue c;
    int num, ch;
    while (1)
    {
        system("cls");
        cout << "*** Circular Queue ***" << endl;
        cout << "1. Enqueue Element" << endl;
        cout << "2. Dequeue Element" << endl;
        cout << "3. Peek Front Operation" << endl;
        cout << "4. Peek Rear Operation" << endl;
        cout << "5. Display the Queue" << endl;
        cout << "6. Exit" << endl
            << endl;
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            cout << "Enter a Value : ";
            cin >> num;
```

```
    c.Enqueue(num);
    getch();
    break;
case 2:
    c.Dequeue();
    getch();
    break;
case 3:
    c.PeekFront();
    getch();
    break;
case 4:
    c.PeekRear();
    getch();
    break;
case 5:
    c.Display();
    getch();
    break;
case 6:
    exit(1);
default:
```

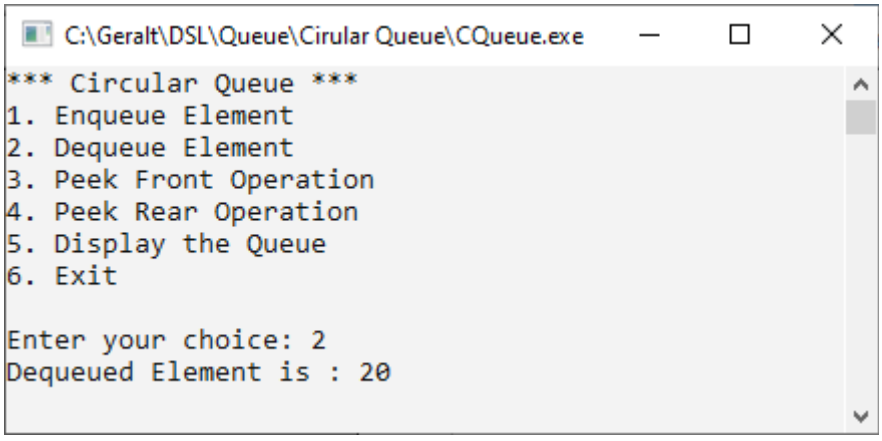
```
        cout << "Incorrect choice !";
        getch();
        break;
        cout << endl;
      } // end of switch
   }    // end of while
} // end of main
```

Output:

Enqueue:



Dequeue:



MET Institute of Computer Science

PeekFront:

```
C:\Geralt\DSL\Queue\Cirular Queue\CQueue.exe
*** Circular Queue ***
1. Enqueue Element
2. Dequeue Element
3. Peek Front Operation
4. Peek Rear Operation
5. Display the Queue
6. Exit

Enter your choice: 3
Element at the Front is 22
```

PeekRear:

```
C:\Geralt\DSL\Queue\Cirular Queue\CQueue.exe
*** Circular Queue ***
1. Enqueue Element
2. Dequeue Element
3. Peek Front Operation
4. Peek Rear Operation
5. Display the Queue
6. Exit

Enter your choice: 3
Element at the Front is 22
```

Display:

```
C:\Geralt\DSL\Queue\Cirular Queue\CQueue.exe
*** Circular Queue ***
1. Enqueue Element
2. Dequeue Element
3. Peek Front Operation
4. Peek Rear Operation
5. Display the Queue
6. Exit

Enter your choice: 5
22 14 16 18
```

MET Institute of Computer Science

Double-Ended Queue:

Code:

```
#include <iostream>
#include <conio.h>
using namespace std;

// 1. Node Template
class DEQNode
{
        public:
        int data;
        DEQNode *right;
        DEQNode *left;
};

// 2. Queues Template
class DEQueue
{
    DEQNode *front;
    DEQNode *rear;
```

```
public:
    DEQueue()
    {
        front = NULL;
        rear = NULL;
    }
    void EnqueueFront(int x);
    void EnqueueRear(int x);
    void DequeueFront();
    void DequeueRear();
    void PeekFront();
    void PeekRear();
    void Display();
    int Empty();
};

// 3. Functions
int DEQueue ::Empty()
{
    int sig = (front == NULL && rear == NULL)
                ? 1
                : 0;
```

```
    return sig;
}
void DEQueue ::EnqueueFront(int x)
{
    DEQNode *t = new DEQNode();
    t->data = x;
    t->right = NULL;
    t->left = NULL;
    if (front == NULL)
    {
        front = t;
        rear = t;
    }
    else
    {
        t->right = front;
        front->left = t;
        front = t;
    }
}
void DEQueue ::EnqueueRear(int x)
{
```

```
    DEQNode *t = new DEQNode();
    t->data = x;
    t->right = NULL;
    t->left = NULL;
    if (front == NULL)
    {
        front = t;
        rear = t;
    }
    else
    {
        t->left = rear;
        rear->right = t;
        rear = t;
    }
}
void DEQueue ::DequeueFront()
{
    if (Empty())
    {
        cout << "Queue Underflow" << endl;
        return;
```

```
    }
    DEQNode *tmp = front;
    if (front == rear)
    {
        front = rear = NULL;
    }
    else
    {
        front = front->right;
        front->left = NULL;
    }
    cout << "Dequeued Element from front " << tmp->data;
    delete tmp;
}
void DEQueue ::DequeueRear()
{
    if (Empty())
    {
        cout << "Queue Underflow" << endl;
        return;
    }
    DEQNode *tmp = rear;
```

```
    if (front == rear)
    {
        front = rear = NULL;
    }
    else
    {
        rear = rear->left;
        rear->right = NULL;
    }
    cout << "Dequeued Element from front " << tmp->data;
    delete tmp;
}
void DEQueue ::Display()
{
    DEQNode *tmp = front;
    if (Empty())
    {
        cout << "Queue Underflow";
        return;
    }
    while(tmp != NULL)
    {
```

```
        cout<<tmp->data<<" ";
        tmp = tmp->right;
        }
}
void DEQueue ::PeekFront()
{
   if (Empty())
   {
      cout << "Queue Underflow!!";
      return;
   }
   DEQNode *tmp = front;
   cout << "Element at the Front is " << tmp->data;
}
void DEQueue ::PeekRear()
{
   if (Empty())
   {
      cout << "Queue Underflow!!";
      return;
   }
   DEQNode *tmp = rear;
```

```
    cout << "Element at the Rear is " << tmp->data;
} // 4. Menu
int main()
{
    DEQueue q;
    int num, ch;
    while (1)
    {
        system("cls");
        cout << "*** Double Ended Queue ***" << endl;
        cout << "1. Enqueue Front Element" << endl;
        cout << "2. Enqueue Rear Element" << endl;
        cout << "3. Dequeue Front Element" << endl;
        cout << "4. Dequeue Rear Element" << endl;
        cout << "5. Peek Front Operation" << endl;
        cout << "6. Peek Rear Operation" << endl;
        cout << "7. Display the Queue" << endl;
        cout << "8. Exit" << endl
            << endl;
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
```

```
{
case 1:
    cout << "Enter a Value : ";
    cin >> num;
    q.EnqueueFront(num);
    getch();
    break;
case 2:
    cout << "Enter a Value : ";
    cin >> num;
    q.EnqueueRear(num);
    getch();
    break;
case 3:
    q.DequeueFront();
    getch();
    break;
case 4:
    q.DequeueRear();
    getch();
    break;
case 5:
```

```
        q.PeekFront();
        getch();
        break;
    case 6:
        q.PeekRear();
        getch();
        break;
    case 7:
        q.Display();
        getch();
        break;
    case 8:
        exit(1);
    default:
        cout << "Incorrect choice !";
        getch();
        break;
        cout << endl;
    } // end of switch
  }    // end of while
} // end of main
```

Output:

Enqueue front:

```
C:\Geralt\DSL\Queue\Double ended Queue\DQueue.exe
*** Double Ended Queue ***
1. Enqueue Front Element
2. Enqueue Rear Element
3. Dequeue Front Element
4. Dequeue Rear Element
5. Peek Front Operation
6. Peek Rear Operation
7. Display the Queue
8. Exit

Enter your choice: 1
Enter a Value : 20
```

Enqueue rear:

```
C:\Geralt\DSL\Queue\Double ended Queue\DQueue.exe
*** Double Ended Queue ***
1. Enqueue Front Element
2. Enqueue Rear Element
3. Dequeue Front Element
4. Dequeue Rear Element
5. Peek Front Operation
6. Peek Rear Operation
7. Display the Queue
8. Exit

Enter your choice: 2
Enter a Value : 40
```

Dequeue front:

```
C:\Geralt\DSL\Queue\Double ended Queue\DQueue.exe
*** Double Ended Queue ***
1. Enqueue Front Element
2. Enqueue Rear Element
3. Dequeue Front Element
4. Dequeue Rear Element
5. Peek Front Operation
6. Peek Rear Operation
7. Display the Queue
8. Exit

Enter your choice: 3
Dequeued Element from front 20
```

Dequeue rear:

```
C:\Geralt\DSL\Queue\Double ended Queue\DQueue.exe
*** Double Ended Queue ***
1. Enqueue Front Element
2. Enqueue Rear Element
3. Dequeue Front Element
4. Dequeue Rear Element
5. Peek Front Operation
6. Peek Rear Operation
7. Display the Queue
8. Exit

Enter your choice: 4
Dequeued Element from front 40
```

MET Institute of Computer Science

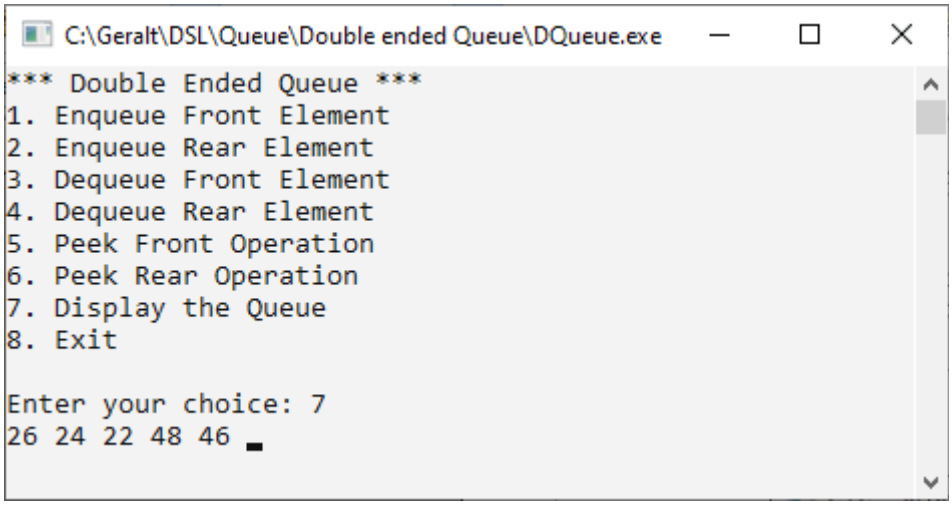
PeekFront:

```
C:\Geralt\DSL\Queue\Double ended Queue\DQueue.exe
*** Double Ended Queue ***
1. Enqueue Front Element
2. Enqueue Rear Element
3. Dequeue Front Element
4. Dequeue Rear Element
5. Peek Front Operation
6. Peek Rear Operation
7. Display the Queue
8. Exit

Enter your choice: 5
Element at the Front is 22
```

PeekRear:

```
C:\Geralt\DSL\Queue\Double ended Queue\DQueue.exe
*** Double Ended Queue ***
1. Enqueue Front Element
2. Enqueue Rear Element
3. Dequeue Front Element
4. Dequeue Rear Element
5. Peek Front Operation
6. Peek Rear Operation
7. Display the Queue
8. Exit

Enter your choice: 6
Element at the Rear is 48
```

Display:



Priority Queue:

CODE:

```
#include<iostream>

#include<conio.h>


using namespace std;


//1. Node template

class PNode

{

        public:

                int data;

                int priority;

                PNode *next;
```

```
};

//2. Queue template

class PQueue
{
	PNode *front;
	public:
		PQueue()
		{
			front = NULL;
		}
		void Enqueue(int x, int p);
		void Dqueue();
		void PeekFront();
		void PeekRear();
		void Display();
};

//3. Functions

void PQueue::Enqueue(int x, int p)
```

```
{
        PNode *t = new PNode();
        t->data = x;
        t->priority = p;
        t->next = NULL;

        //first node
        if(front == NULL)
        {
                front = t;
                return;
        }
        //Traverse in order
        PNode *tmp = front;
        PNode *prev = NULL;

        while(tmp != NULL && tmp->priority < t->priority)
        {
                prev = tmp;
                tmp = tmp->next;
        }
```

```
        //Insert t at the correct position in the queue


        if(tmp == front) //Front node Insertion
        {
                t->next = front;
                front = t;
        }
        else if(tmp == NULL) //Last node insertion
        {
                prev->next = t;
        }
}

void PQueue::Dqueue()
{
        PNode *tmp = front;
        if(front->next == NULL)
        {
                front == NULL;
        }
        else
        {
```

```
        front = front->next;
    }
    cout<<"Element "<<tmp->data<<" dequeued with priority "<<tmp->priority;
    delete tmp;
}

void PQueue::PeekFront()
{
    PNode *tmp = front;
    cout<<"Element at the front is "<<tmp->data<<" with priority "<<tmp->priority;
}

void PQueue::PeekRear()
{
    PNode *tmp = front;
    while(tmp->next != NULL)
    {
        tmp = tmp->next;
    }
    cout<<"Element at the front is "<<tmp->data<<" with priority "<<tmp->priority;
}
```

```
void PQueue::Display()
{
	if(front == NULL)
	{
		cout<<"Empty Queue";
		return;
	}
	PNode *tmp = front;
	cout<<"Data | Priority \n";

	while(tmp != NULL)
	{
		cout<<tmp->data<<" | "<<tmp->priority<<"\n";
		tmp = tmp->next;
	}
}

//4. Menu

int main()
{
	int ch, num, pri;
```

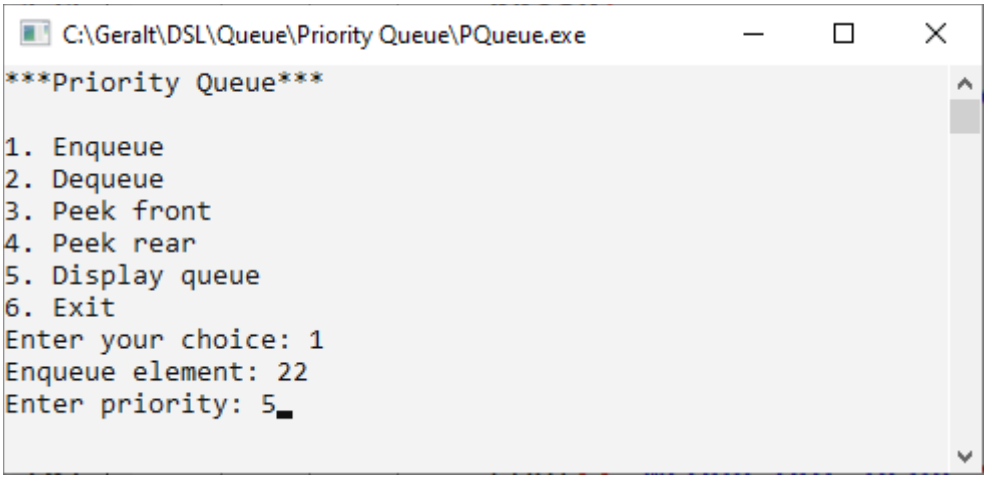
```
PQueue p;

while(1)
{
        system("cls");
        cout<<"***Priority Queue*** \n\n";

        cout<<"1. Enqueue \n";
        cout<<"2. Dequeue \n";
        cout<<"3. Peek front \n";
        cout<<"4. Peek rear \n";
        cout<<"5. Display queue \n";
        cout<<"6. Exit \n";

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

        switch(ch)
        {
                case 1:
                        cout<<"Enqueue element: ";
                        cin>>num;
```

```
        cout<<"Enter priority: ";
        cin>>pri;
        p.Enqueue(num,pri);
        getch();
        break;
case 2:
        cout<<"Dequeue element: ";
        p.Dqueue();
        getch();
        break;
case 3:
        cout<<"Peek front: ";
        p.PeekFront();
        getch();
        break;
case 4:
        cout<<"Peek rear: ";
        p.PeekRear();
        getch();
        break;
case 5:
        cout<<"Display queue: \n";
```

```
                p.Display();
                getch();
                break;
            case 6:
                exit(1);
                break;
            default:
                cout<<"Wrong opt bruv";
                getch();
                break;
        }//end switch
    }//end while
}//end main
```

OUTPUT:

ADD ELEMENT:



Display:

```
C:\Geralt\DSL\Queue\Priority Queue\PQueue.exe
***Priority Queue***

1. Enqueue
2. Dequeue
3. Peek front
4. Peek rear
5. Display queue
6. Exit
Enter your choice: 5
Display queue:
Data | Priority
24 | 2
22 | 5
26 | 10
28 | 15
```

PeekFront:

```
C:\Geralt\DSL\Queue\Priority Queue\PQueue.exe
***Priority Queue***

1. Enqueue
2. Dequeue
3. Peek front
4. Peek rear
5. Display queue
6. Exit
Enter your choice: 3
Peek front: Element at the front is 24 with priority 2
```

PeekRear:

```
C:\Geralt\DSL\Queue\Priority Queue\PQueue.exe
***Priority Queue***

1. Enqueue
2. Dequeue
3. Peek front
4. Peek rear
5. Display queue
6. Exit
Enter your choice: 4
Peek rear: Element at the front is 28 with priority 15
```

Dequeue:

```
C:\Geralt\DSL\Queue\Priority Queue\PQueue.exe
***Priority Queue***

1. Enqueue
2. Dequeue
3. Peek front
4. Peek rear
5. Display queue
6. Exit
Enter your choice: 2
Dequeue element: Element 24 dequeued with priority 2
```