|  |  |
| --- | --- |
| Program No | 16 |
| Roll No | 1333 |
| Unit | 04 |
| Program | Ordinary Queues |

**/\* Name: Harshada naik Roll no.: 1333**

**unit 4: Queues**

**program: Ordinary Queue \*/**

#include<iostream> #include<conio.h> #define MAX 5

using namespace std;

/\* 1. Node Template - Not required \*/

/\* 2. Queue Template \*/ class OQueue

{

int a[MAX]; int front, rear;

public:

OQueue()

{

front = -1;

rear = -1;

}

void Enqueue(int x); void Dequeue(); void PeekFront(); void PeekRear(); void Display();

int Full(); int Empty();

};

/\* 3. Functions \*/

//Enqueue

void OQueue :: Enqueue(int x)

{

if(Full())

{

cout << "Queue Overflow!"; return;

}

if(front == -1) //first element in the queue

{

front++;

}

rear++; a[rear] = x;

}

int OQueue :: Full()

{

return (rear == MAX-1?1:0);

}

//Dequeue

void OQueue :: Dequeue()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

int t = a[front];

if(front == rear) //single element

{

front = -1;

rear = -1;

}

else

{

front++;

}

cout << "Element removed is: " << t;

}

int OQueue :: Empty()

{

return (front == -1?1:0);

}

//Peekfront

void OQueue :: PeekFront()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

cout << "Element at front is: " << a[front];

}

//PeekRear

void OQueue :: PeekRear()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

cout << "Element at end is: " << a[rear];

}

//Display

void OQueue :: Display()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

for (int i=front; i<=rear; i++)

{

cout << a[i] << " ";

}

}

/\* 4. menu \*/ int main()

{

int ch, num; OQueue q;

while(1)

{

system("cls");

cout <<"\*\*\*Ordinary Queue \*\*\*\n\n";

cout << "1. Enqueue an element" << endl; cout << "2. Dequeue" << endl;

cout << "3. Peek Front" << endl; cout << "4. Peek Rear" << endl;

cout << "5. Display the queue" << endl; cout << "6. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter an element: "; cin >> num; q.Enqueue(num);

getch(); break;

case 2:

q.Dequeue();

getch(); break;

case 3:

q.PeekFront();

getch(); break;

case 4:

q.PeekRear();

getch(); break;

case 5:

q.Display();

getch(); break;

case 6:

exit(1);

default:

cout << "incorrect choice!"; getch();

break;

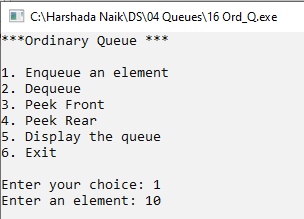
}//end of switch

}//end of while

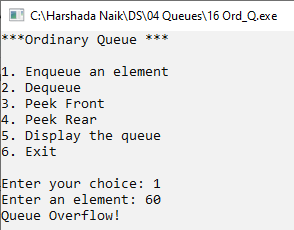
} //end of main

**Output:**

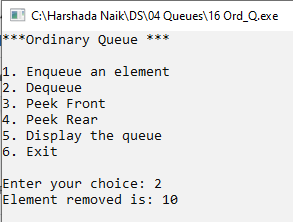
**Enqueue:**



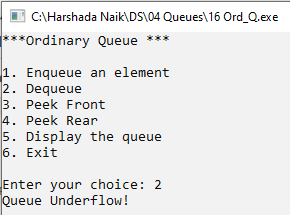
**When queue exceeds its limit:**



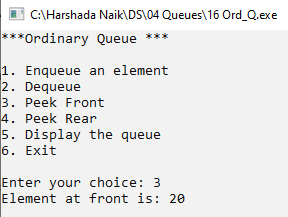
**Dequeue:**



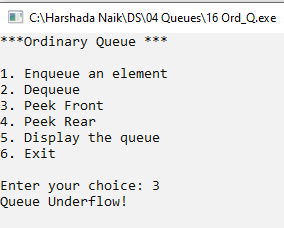
**When queue is empty:**



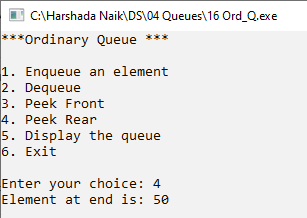
**Peek front:**



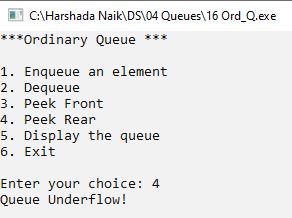
**When queue is empty:**



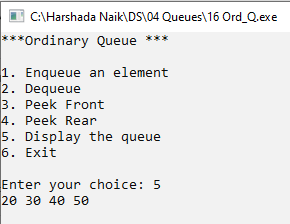
**PeekRear:**



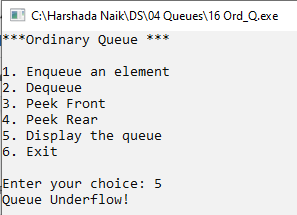
**When queue is empty:**



**Display:**



**When queue is empty:**



|  |  |
| --- | --- |
| Program No | 17 |
| Roll No | 1333 |
| Unit | 04 |
| Program | Circular Queues |

**/\* Name: Harshada naik Roll no.: 1333**

**unit 4: Queues**

**program: Circular Queue \*/**

#include<iostream> #include<conio.h> #define MAX 5

using namespace std;

/\* 1. Node template = not required \*/

/\* 2. Queue Template \*/ class CQueue

{

int a[MAX]; int front, rear; int cnt;

public:

CQueue()

{

front = rear = -1; cnt = 0;

}

void Enqueue(int x); void Dequeue(); void PeekFront(); void PeekRear(); void Display();

int Full(); int Empty();

};

/\* 3. functions \*/

//Enqueue

void CQueue :: Enqueue(int x)

{

if (Full())

{

cout << "Queue Overflow!"; return;

}

if(front == -1) //first element in the list

{

front++;

}

rear = (rear + 1)%MAX; a[rear] = x;

cnt++;

}

int CQueue :: Full()

{

return(cnt == MAX?1:0);

}

//Dequeue

void CQueue :: Dequeue()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

int t = a[front];

if(front == rear) //single element

{

front = -1;

rear = -1;

}

else

{

front = (front+1)%MAX;

}

cout << "Element removed is: " << t; cnt--;

}

int CQueue :: Empty()

{

return(cnt == 0?1:0);

}

//Peekfront

void CQueue :: PeekFront()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

cout << "Element at front is: " << a[front];

}

//PeekRear

void CQueue :: PeekRear()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

cout << "Element at end is: " << a[rear];

}

//Display

void CQueue :: Display()

{

if(Empty())

{

cout << "Queue Underflow!"; return;

}

int x, i = front;

for(x=1; x<=cnt; x++)

{

cout << a[i] << " "; i = (i+1)%MAX;

}

}

/\* 4. Menu \*/ int main()

{

int num, ch; CQueue c;

while(1)

{

system("cls");

cout << "\*\*\*Circular Queue\*\*\*\n\n";

cout << "1. Enqueue an element" << endl; cout << "2. Dequeue" << endl;

cout << "3. Peek Front" << endl; cout << "4. Peek Rear" << endl;

cout << "5. Display the queue" << endl; cout << "6. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter an element: "; 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;

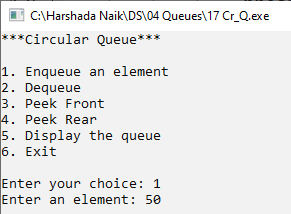
}//end of switch

}//end of while

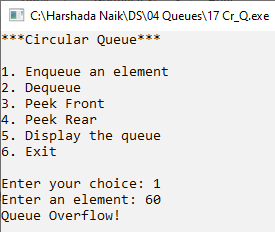
}//end of main

**Output:**

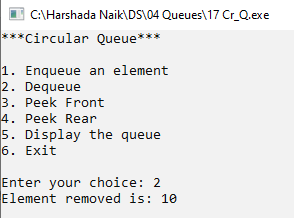
**Enqueue:**



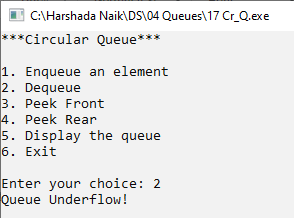
**When queue exceeds limit:**



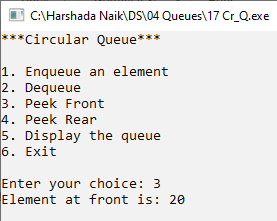
**Dequeue:**



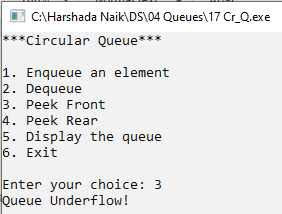
**When queue is empty:**



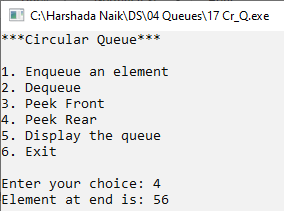
**Peekfront**:



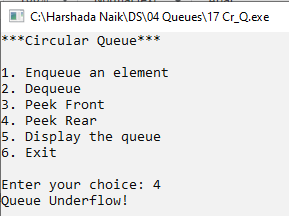
**When queue is Empty:**



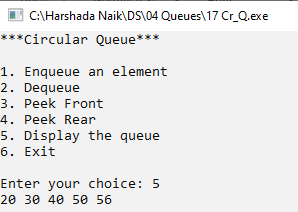
**Peekrear:**



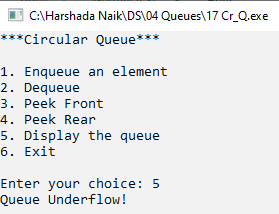
**When queue is empty:**



**Display the circularity after dequeue and enqueue:**



**When queue is empty:**



|  |  |
| --- | --- |
| Program No | 18 |
| Roll No | 1333 |
| Unit | 04 |
| Program | Double-Ended Queues |

**/\* Name: Harshada naik Roll no.: 1333**

**unit 4: Queues**

**program: Double-ended Queue \*/**

#include<iostream> #include<conio.h>

using namespace std;

/\* 1. Node Template \*/ class DQNode

{

public:

int data;

DQNode \*right;

DQNode \*left;

};

/\* 2. Double-ended Queue Template \*/ class DQueue

{

DQNode \*front;

DQNode \*rear;

public:

DQueue()

{

front = NULL; rear = NULL;

}

void EnqueueFront(int x); void EnqueueRear(int x); void DequeueFront(); void DequeueRear();

void PeekFront(); void PeekRear(); void Display();

};

/\* 3. Functions \*/

//Enqueue front

void DQueue :: EnqueueFront(int x)

{

//make a node

DQNode \*t = new DQNode(); t->data = x;

t->right = NULL; t->left = NULL;

//Special case- First Node if(front == NULL)

{

front = t; rear = t;

}

else //Any other node

{

t->right = front; front->left = t; front = t;

}

}

//Enqueue Rear

void DQueue :: EnqueueRear(int x)

{

//make a node

DQNode \*t = new DQNode(); t->data = x;

t->left = NULL; t->right = NULL;

//special case- First Node if(front == NULL)

{

front = t; rear = t;

}

else //Any other node

{

rear->right = t; t->left = rear; rear = t;

}

}

//Dequeue Front

void DQueue :: DequeueFront()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

DQNode \*tmp = front;

if(front == rear) //single node deletion

{

front = NULL; rear = NULL;

}

else //Any other node

{

front = front->right; front->left = NULL;

}

cout << "Element removed: " << tmp->data;

delete tmp;

}

//Dequeue Rear

void DQueue :: DequeueRear()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

DQNode \*tmp = rear;

if(front == rear) //Single node deletion

{

front = NULL; rear = NULL;

}

else //any other node

{

rear = rear->left; rear->right = NULL;

}

cout << "Element removed: " << tmp->data; delete tmp;

}

//Peekfront

void DQueue :: PeekFront()

{

if(front == NULL)

{

cout << "Queue Underflow"; return;

}

cout << "Element at front is: " << front->data;

}

//PeekRear

void DQueue :: PeekRear()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

cout << "Element at rear is: " << rear->data;

}

//Display

void DQueue :: Display()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

DQNode \*tmp = front;

while(tmp!=NULL)

{

cout << tmp->data << " <-> "; tmp = tmp->right;

}

cout << "NULL";

}

/\* 4. Menu \*/ int main()

{

int ch, num; DQueue d;

while(1)

{

system("cls");

cout << "\*\*\*Double-Ended Queue\*\*\*\n\n";

cout << "1. Enqueue Front" << endl; cout << "2. Enqueue Rear" << endl; cout << "3. Dequeue Front" << endl; cout << "4. Dequeue Rear" << endl;

cout << "5. peek Front" << endl; cout << "6. Peek Rear" << endl;

cout << "7. Display the Queue" << endl; cout << "8. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter an element: "; cin >> num; d.EnqueueFront(num);

getch(); break;

case 2:

cout << "Enter an element: "; cin >> num; d.EnqueueRear(num);

getch(); break;

case 3:

d.DequeueFront();

getch(); break;

case 4:

d.DequeueRear();

getch(); break;

case 5:

d.PeekFront();

getch(); break;

case 6:

d.PeekRear();

getch(); break;

case 7:

d.Display();

getch(); break;

case 8:

exit(1);

default:

cout << "Incorrect option";

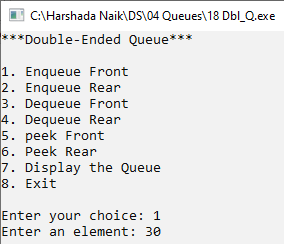
getch(); break;

}//end of switch

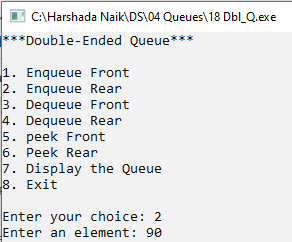
}//end of while

}//end of main

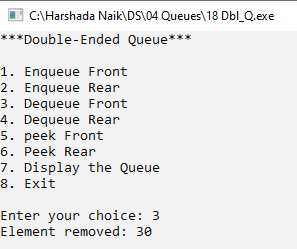
**Output: EnqueueFront:**



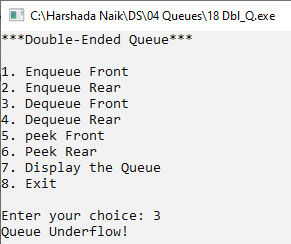
**EnqueueRear:**



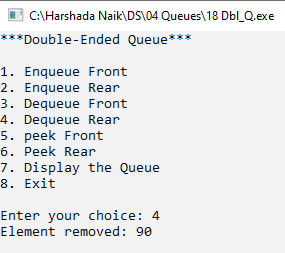
**DequeueFront:**



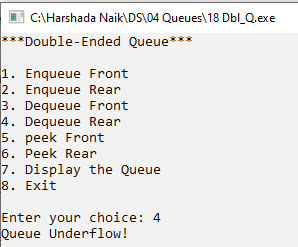
**When Queue is Empty:**



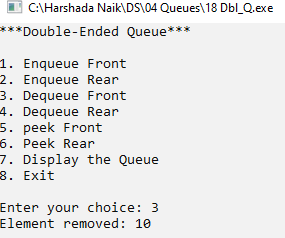
**DequeueRear:**



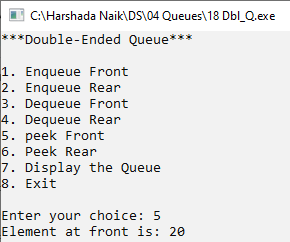
**When Queue is Empty:**



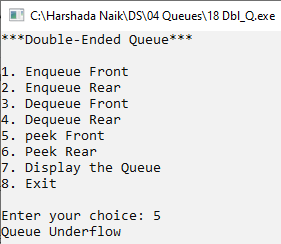
**Single node deletion:**



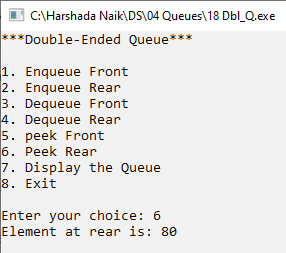
**PeekFront:**



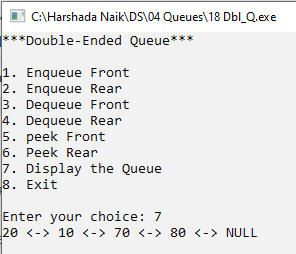
**When Queue is empty:**



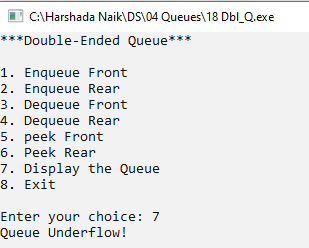
**PeekRear:**



**Display:**



**When Queue is Empty:**



|  |  |
| --- | --- |
| Program No | 19 |
| Roll No | 1333 |
| Unit | 04 |
| Program | Priority Queues |

**/\* Name: Harshada naik Roll no.: 1333**

**unit 4: Queues**

**program: Priority Queue \*/**

#include<iostream> #include<conio.h>

using namespace std;

/\* 1. Node Template \*/ class PQNode

{

public:

int data; int priority;

PQNode \*next;

};

/\* 2. Priority Queue Template \*/ class PQueue

{

PQNode \*front; PQNode \*rear;

public:

PQueue()

{

front = NULL; rear = NULL;

}

void Enqueue(int x, int p); void Dequeue();

void PeekFront(); void PeekRear(); void Display();

};

/\* 3. Function \*/

//Enqueue

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

{

//Make a node

PQNode \*t = new PQNode(); t->data = x;

t->priority = p; t->next = NULL;

//First node in the queue if(front == NULL)

{

front = t; return;

}

//Ordered Traversal PQNode \*tmp = front; PQNode \*prev;

while(tmp!=NULL && tmp->priority<p)

{

prev = tmp;

tmp = tmp->next;

}

//insert t in the queue in order of priority if(tmp == front) //first node

{

t->next = front; front = t;

}

else if(tmp == NULL) //Last Node

{

prev->next = t;

}

else //Middle node

{

prev->next = t; t->next = tmp;

}

}

//Dequeue

void PQueue :: Dequeue()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

PQNode \*tmp = front;

if(front->next == NULL) //Single node deletion

{

front = NULL;

}

else

{

front = front->next;

}

cout << "Element Removed: " << tmp->data; cout << " with Priority: " << tmp->priority;

delete tmp;

}

//PeekFront

void PQueue :: PeekFront()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

cout << "Element at front is: " << front->data << "\nPriority at front is: " << front->priority;

}

//PeekRear

void PQueue :: PeekRear()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

PQNode \*tmp = front;

while(tmp->next!=NULL)

{

tmp = tmp->next;

}

cout << "Element at rear is: " << tmp->data << "\nPriority at rear is: " << tmp->priority;

}

//Display

void PQueue :: Display()

{

if(front == NULL)

{

cout << "Queue Underflow!"; return;

}

PQNode \*tmp = front;

while(tmp!=NULL)

{

cout << "Data: " << tmp->data << " Priority: " << tmp->priority << " -> " << endl; tmp = tmp->next;

}

cout << "NULL";

}

/\* 4. Menu \*/ int main()

{

int num, ch, pr;

PQueue p;

while(1)

{

system("cls");

cout << "\*\*\*Priority Queue\*\*\*\n\n";

cout << "1. Enqueue" << endl; cout << "2. Dequeue" << endl; cout << "3. Peek Front" << endl; cout << "4. Peek Rear" << endl;

cout << "5. Display the Queue" << endl; cout << "6. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter an element: "; cin >> num;

cout << "Enter its Priority: "; cin >> pr; p.Enqueue(num,pr);

getch(); break;

case 2:

p.Dequeue();

getch(); break;

case 3:

p.PeekFront();

getch(); break;

case 4:

p.PeekRear();

getch(); break;

case 5:

p.Display();

getch(); break;

case 6:

exit(1);

default:

cout << "Incorrect Option"; getch();

break;

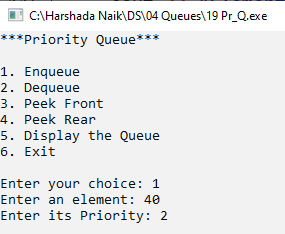
}//end of switch

}//end of while

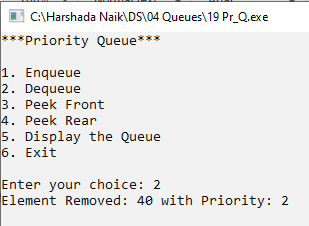
}//end of main

**Output:**

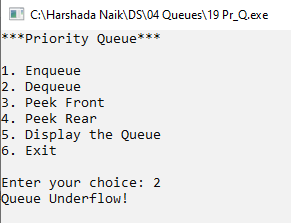
**Enqueue:**



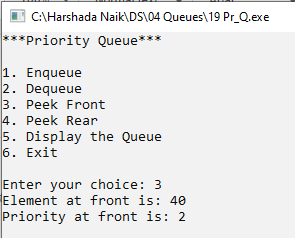
**Dequeue:**



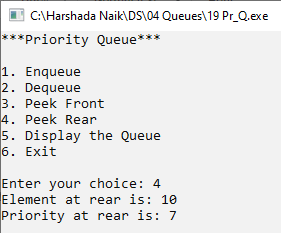
**When Queue is Empty:**



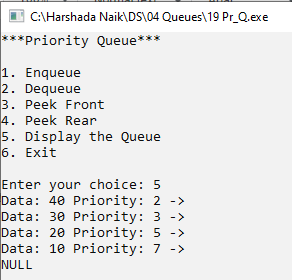
**PeekFront:**



**Peekrear:**



**Display:**



**When Queue is Empty:**

