PRN No. : 124B2B012

Name : Khairnar Atharva Anil.

Title: Implement a job scheduling system for a manufacturing plant using a double-ended queue. The system needs to efficiently manage the processing of jobs on various machines throughout the plant. Each job has a Job_priority. The system should support the following operations:
a. Add Job
b. Remove Job
c. Display Job
d. Search Job

Code:

```
#include <iostream>


class Node
{
    public:
    int data;
    int priority;
    Node *next;
    Node *prev;

    public: Node(int data1, int priority1)
    {
        next=prev=NULL;
        data=data1;
        priority=priority1;
    }
};

class Dequeue
{
    Node *front,*rear;
    public: Dequeue()
    {
        front=rear=NULL;
    }
   public:
   void insertfromr(int data,int priority)
   {
       Node *nn=new Node(data,priority);
       if(rear==nullptr)
```

```
   {
      front=rear=nn;
    }
    else
    {
      if(nn->priority>=rear->priority)
      {
         rear->next=nn;
         nn->prev=rear;
         rear=nn;
      }

      else
      {
         Node *temp=rear;
         while(temp!=nullptr && temp->priority > nn->priority)
         {
            temp=temp->prev;
         }
         if (temp == nullptr) {
            nn->next = front;
            front->prev = nn;
            front = nn;
         }
         else
         {
            nn->next=temp->next;
            nn->prev=temp;
            if(temp->next!=NULL)
            {
               temp->next->prev=nn;
            }
            temp->next=nn;
         }
      }
    }
 }

 void insertfromf(int data1,int priority1)
{
   Node *nn=new Node(data1,priority1);
   if(front==nullptr)
   {
```

```
        front=rear=nn;
    }
    else
    {
        if(nn->priority<=front->priority)
        {
            nn->next = front;
            front->prev = nn;
            front=nn;
        }
        else
        {
            Node *temp=front;
            while(temp->next != nullptr && temp->priority < nn->priority)
            {
                temp=temp->next;
            }
            if(temp==rear)
            {
                rear->next=nn;
                nn->prev=rear;
                rear=nn;
            }
            else
            {
                nn->next=temp;
                nn->prev=temp->prev;
                if(temp->prev!=NULL)
                {
                    temp->prev->next = nn;
                }
                temp->prev = nn;
            }
        }
    }
}

void display()
{
    Node *temp = front;
    while (temp != NULL)
    {
        std::cout << temp->data << " ";
```

```
            temp = temp->next;
        }
        std::cout << std::endl;
    }
};

int main()
{
    Dequeue d;
    d.insertfromr(10,2);
    d.insertfromr(11,1);
    d.insertfromf(5,3);
    d.display();
}
```

Output: