Write a program to create a priority queue and perform the following.

Print the queue elements.

Print the top element.

Print the queue after removing the top element.

Print the status of the searched element.

Convert the queue into an object array and print the elements

```
import java.io.*;
import java.util.*;
class Main {
public static void main(String [] args) {
                        int i,n;
                        Scanner sc = new Scanner(System.in);
                        n =Integer.parseInt(sc.nextLine());
                        PriorityQueue<String> pQueue = new PriorityQueue<String>(n);
                        for(i=0;i<n;i++) {
                                pQueue.add(sc.nextLine());
                        }
                        String search = sc.nextLine();
                        Iterator itr = pQueue.iterator();
                        while (itr.hasNext())
                                System.out.print(itr.next()+" ");
                        System.out.println();
                        System.out.println("The first element in the array is "+pQueue.poll());
                        System.out.println("After removing first element");
                        Iterator<String> itr2 = pQueue.iterator();
                        while (itr2.hasNext())
                                System.out.print(itr2.next()+" ");
                        System.out.println();
                        boolean b = pQueue.contains(search);
                        if(b == true) {
                                System.out.println(search+" is present in the array");
```

```
                }
                else {
                        System.out.println(search+" is not present in the array");
                }
                Object[] arr = pQueue.toArray();
                System.out.println("Array elements : ");
                for (i = 0; i < arr.length; i++)
                        System.out.print(arr[i].toString()+" ");
        }
    }
```

---

Mr. John is doing a transport business. As he had a lot of customers for registering the car, he wants to give the preference for the person who comes first, if the car is filled with n number after that who comes with EVEN is added and the first element is removed .If ODD means do nothing.

```
#include <stdio.h>
#include <stdlib.h>
int QUEUE[100],front=-1,rear=-1;
void insert_in_Q(int queue[],int ele, int MAX)
{
   if(rear==-1)
   {
      front=rear=0;
      queue[rear]=ele;
   }
   else
   {
      rear++;
      queue[rear]=ele;
   }
}
void display_Q(int queue[])
{      int i;
   for(i=front;i<=rear;i++)
```

```
    { printf("%d ",queue[i]); }

}

void remove_from_Q(int queue[])
{
    int ele;
    if(front==rear)
    {
        ele=queue[front];
        front=rear=-1;
    }
    else
    {
        ele=queue[front];
        front++;
    }
}
int main()
{
    int ele,i,n1,n2;
    scanf("%d %d",&n1,&n2);
    if ((n1<1)||(n2<1))
    {
    printf("%d",-1);
      return 0;
      }
    for(i=0;i<n2;i++)
    {
    scanf("%d",&ele);
        if ( i>=n1)
        {
```

```
        if((ele%2)==0)
        {
        remove_from_Q(QUEUE);
        insert_in_Q(QUEUE,ele,n2+1);
        }
        }
        else
        {
        insert_in_Q(QUEUE,ele,n2+1);
        }
    }
display_Q(QUEUE);
return 0;
}
```

---

Write a program to insert and delete elements in the priority queue.

Priority starts from 0.

The node with 0 priority is always 1.

```
#include <bits/stdc++.h>
using namespace std;
typedef struct node
{
        int data;
        int priority;

        struct node* next;

} Node;
Node* newNode(int d, int p)
{
        Node* temp = (Node*)malloc(sizeof(Node));
        temp->data = d;
```

```
	temp->priority = p;
	temp->next = NULL;

	return temp;
}
int peek(Node** head)
{
	return (*head)->data;
}
void pop(Node** head)
{
	Node* temp = *head;
	(*head) = (*head)->next;
	free(temp);
}
void push(Node** head, int d, int p)
{
	Node* start = (*head);
	Node* temp = newNode(d, p);
	if ((*head)->priority > p)
	{
		temp->next = *head;
		(*head) = temp;
	}
	else
	{
		while (start->next != NULL &&
			start->next->priority < p)
		{
			start = start->next;
		}
		temp->next = start->next;
```

```
                start->next = temp;
        }
}
int isEmpty(Node** head)
{
        return (*head) == NULL;
}
int main()
{
        Node* pq = newNode(1, 0);
        int i,n;
        cin>>n;
        for(i=0;i<n;i++) {
           int val,p;
           cin>>val>>p;
           push(&pq,val,p);
        }
        while (!isEmpty(&pq))
        {
                cout << " " << peek(&pq);
                pop(&pq);
        }
        return 0;
}
```

---

The customer service executive of an electronics company attends her complaints in an order so that priority is given on first come first serve basis. Every executive of the company is assigned a particular set of token numbers based on the token allocated first for the day.

The system is built up in such a way that the next token does not get unlocked unless the previous one is closed. An engineer who was part of the executive team is trying to understand this system and started building his own for practice.

```
void enQueue(struct node *ptr, int item)
{
   ptr = (struct node *) malloc (sizeof(struct node));
```

```
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
        return;
    }
    else
    {
        ptr -> data = item;
        if(front == NULL)
        {
            front = ptr;
            rear = ptr;
            front -> next = NULL;
            rear -> next = NULL;
        }
        else
        {
            rear -> next = ptr;
            rear = ptr;
            rear->next = NULL;
        }
    }
}
 void deQueue (struct node *ptr)
{
    if(front == NULL)
    {
        printf("\nUNDERFLOW\n");
        return;
    }
    else
    {
```

```
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}
void display(struct node *ptr)
{
    ptr = front;
    if(front == NULL)
    {
        printf("\nEmpty queue\n");
    }
    else
    {
        while(ptr != NULL)
        {
            printf("%d ",ptr -> data);
            ptr = ptr -> next;
        }
    }
}
```

---

Hitler decided to declare war on Americans so he decided to build a gun on their own.These guns consist of N pieces of Steel of different length. These must be joined in order to create the gun. The cost to connect two pieces is equal to some of their lengths. To help Hitler write a program that can calculate the lowest possible cost to build the Gun.

```
void sort_Queue(int *queue)
{
int a,j,i;
for(i=front;i<=rear;i++)
{
    for(j=front+i;j<=rear;j++)
    {
```

```
        if (*(queue+i) > *(queue+j))
        {

            a =  *(queue+i);
            *(queue+i) = *(queue+j);
            *(queue+j) = a;

        }
    }
}

 int *b=(int*)malloc(sizeof(int)*rear);
 int sum=*(queue+0);
int c=0;
for(i=1;i<=rear;i++)
{
sum=sum+*(queue+i);
*(b+i-1)=sum;
c=c+1;
}
sum=0;
for(i=0;i<c;i++)
sum=sum+*(b+i);

printf("%d",sum);
    }
```