

DSA Lab

Name: Siddhi Parekh

Reg No.: 221071047

Batch: C

SY Comps

Experiment No.: 6

AIM:-

To implement Maximum Priority queues using a Linked List.

THEORY:-

The principle of priority queues finds various applications especially in tech and software fields. It can be summarized by giving an example. when there are multiple application downloads

At that time the task with the most weight i.e. of highest priority is executed first compared to others based on priority, after which other applications download proceed. Priority helps us control flow of how things are executed.

In our experiment we have implemented it with the help of a Linked List. Linked List is a form of Physical data structure just like Array whereas Queue is a logical data structure.

This unique combination of data structures has been used to implement Priority queues.

Application wise Linked List tells us what our next procedure is. This can be commonly seen while operating on a webpage. When you click a button it redirects you to a new page as that

button contains the URL of that next page integrated to it. The way in which there is URL which helps us to navigate in a similar way we have address in each node of linked-list which helps us

to navigate to the next element hence navigating through the entire linked-list.

ALGORITHM:

Step 0: Start

Step 1: Create a structure which represents node of a list containing the value
And address of the next node in the current node.

Step 2: Create(int n)

```
for(int i=0; i<n; i++)
```

```
    Struct node*p,*q,*r
```

```
    r=getnode()
```

```
    if(start==NULL)
```

```
        Insert the node at beginning of the list
```

```
    else
```

```
        while(p!=NULL)
```

```
            if(r < q->data && r>data>p->data) insert in between them
```

```
            else if(r->data>q->data && q==start) insert at beginning
```

```
            else iterate the list
```

```
        if(r->data<q->data && p==NULL)insert at the end
```

```
        else if(r->data>q->data && p==NULL) insert before q, if q only exists in  
            The list.
```

Step 3: Display the elements of the list

Step 4: Inside the int main call create and display functions.

Step 5: Stop

EXAMPLE:-

Let the numbers inserted into the linked list be in the sequence:10,40,30,20.

So,the insert function arranges this sequence as:



Upon calling the 'display' function, it will output the values as 10,20,30,40 as this is the sequence of the data in the linked list. Now since the linked list is in a sorted manner, when we call the 'Max' function, it will return the last element of the linked list. In order to delete the maximum value, we call 'Extract_Max' function which will delete the last value from the linked list and now the maximum value becomes 30. Hence, this linked list works in this fashion as a priority queue.

CONCLUSION:-

In this experiment we have understood how priority queues can be effectively used as one of the ways to sort elements (ascending or descending order). The code helps us to explain how priority queues (a logical data structure) can use a linked-list (a physical data structure). To conclude we can say that here we have inserted elements in a linked-list just in a way of how priority queues work.

CODE:

```
#include<iostream>
using namespace std;

struct node
{
    int data;
    struct node * next;
};

struct node * start=NULL;

void create (int n)
{
    for(int i=0; i<n; i++){

        int a;
        node *p, *q,*r;
```

```
r=(node*)malloc(sizeof(node));
```

```
cout<<"Enter value of counter "<<i+1<<endl;  
cin>>a;
```

```
r->data = a;  
r->next = NULL;
```

```
if(start == NULL)  
{  
    start = r;  
    r->next = NULL;  
}
```

```
else  
{
```

```
    q=start;  
    p=q->next;
```

```
while(p!=NULL)  
{  
    if(r->data<q->data && r->data>p->data)  
    {  
        r->next=p;  
        q->next=r;  
        break;  
    }
```

```
    else if(r->data>q->data && q==start)  
    {  
        r->next=q;  
        start=r;
```

```

        break;
    }

    else
    {
        q=p;
        p=p->next;
    }
}

if(r->data<q->data && p==NULL)
{
    r->next=NULL;
    q->next=r;
}

else if(r->data>q->data && p==NULL)
{
    r->next=q;
    start=r;
}

}
}

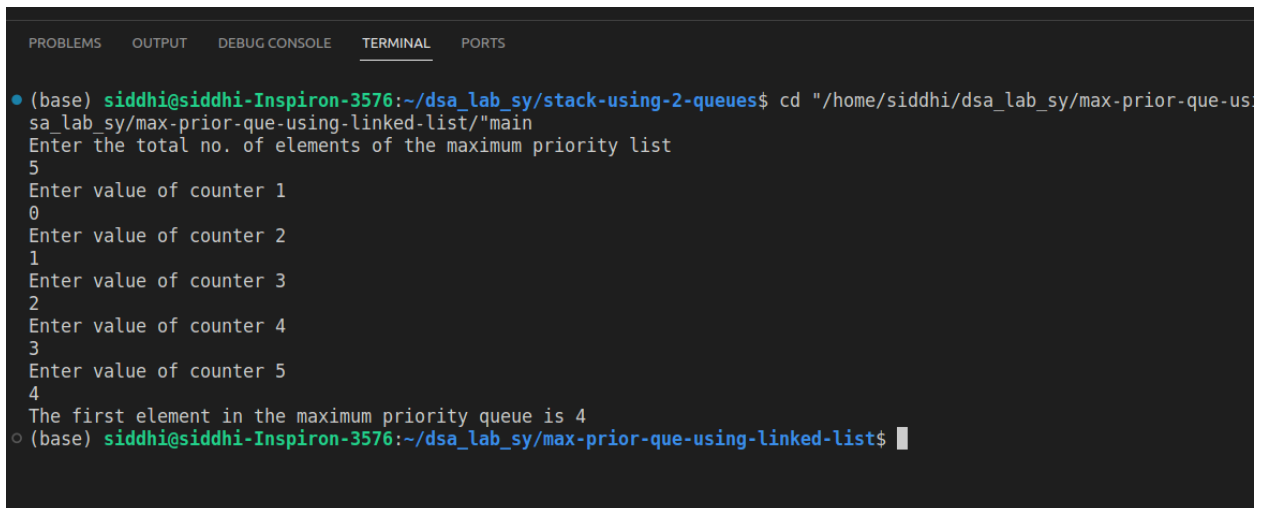
}

void display()
{
    node *m=(node*)malloc(sizeof(node));
    m=start;
    cout<<"The first element in the maximum priority queue is
"<<m->data<<endl;
}

```

```
int main()
{
    int d;
    cout<<"Enter the total no. of elements of the maximum priority list"<<endl;
    cin>>d;
    create(d);
    display();
}
```

OUTPUT:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• (base) siddhi@siddhi-Inspiron-3576:~/dsa_lab_sy/stack-using-2-queues$ cd "/home/siddhi/dsa_lab_sy/max-prior-que-us
sa_lab_sy/max-prior-que-using-linked-list/"main
Enter the total no. of elements of the maximum priority list
5
Enter value of counter 1
0
Enter value of counter 2
1
Enter value of counter 3
2
Enter value of counter 4
3
Enter value of counter 5
4
The first element in the maximum priority queue is 4
○ (base) siddhi@siddhi-Inspiron-3576:~/dsa_lab_sy/max-prior-que-using-linked-list$
```