Assignment No: 11

Aim: Write a C++ program to simulating job queue by using array and linked list.

Objectives:

1. To study queue using array
2. To study queue using linked list.

Problem Statement:

Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and delete job from queue.

Software Requirements:

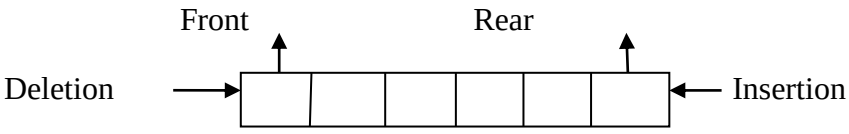
Ubuntu Linux14.04,GCC / G++(editor)

Theory and Concept:

1. Queue:

It is a special kind of list, where items are inserted at one end(the rear) and deleted from the other end(front). Queue is a FIFO(First In First Out) list.

We come across the term queue in our day to day life. We see a queue at a railway reservation counter, or a movie theatre ticket counter. Before getting the service, one has to wait in the queue. After receiving the service, one leaves the queue. Services is provided at one end (the front) and people join the other end(rear).



2. Queue Using Array:

An array representation of queue requires three entities:

a. An array to hold queue elements.
b. A variable to hold the index of the front element.
c. A variable to hold the index of the rear element.

A queue data type may be defined formally as follows:

```
# define MAX 30

Typedef struct queue

{

    int data[MAX];

    int front, rear;

}queue;
```

a. Initialization Of Queue:

```
Void init(struct que *q)
{
    int i;
    for(i=0;i<MAX;i++)
    q->arr[i]=-1;
    q->front=0;
    q->rear=-1;
}
```

b. Print Function:

```
for(i=0;i<MAX;i++)
cout<<" "<<q.arr[i];
cout<<"/t/tFront="<<q.front<<"    rear="<<q.rear;
cout<<"\n\n";
ch=getchar();
```

c. Isempty Condition:

```
Int isempty(que *que.int data)
{
    returnq.rear<q.front?1:0;
}
```

d. Isfull Condition:

```
Int isfull(structque q)
    {
        Return q.rear==MAX-1?1:0;
    }
```

e. Add Function:

Algorithm for insertion in a queue:

1. Insertion in an empty queue.
   rear=front=0;
   data[rear]=x;
2. Inserting in a non-empty queue.
   Rear=rear+1;
   data[rear]=x;

```
void add(structque *q, int data)
    {
        q->rear+=1;
        q->arr [q->rear]=data;
    }
```

f. Delete Function:

Algorithm for deletion from queue:

1. Deletion of last element or only element(rear=front)
   a) x=data[front]
   b) rear=front=-1;
   c) return x;
2. Deletion of the element when the queue length >1
   a) x=data[front]
   b) front=front+1
   c) return x;

3. Queue Using Linked List:

a. Initialization Function:

```
que::que()
    {
        front=rear=NULL;
    }
```

b. Create Function:

```
cout<<"\nEnter the no. of Jobs you want to create:";
cin>>n;
```

```
for(i=0;i<n;i++)
{
        Cout<<"\nEnter the jobs:";
        Cin>>data;
        if(front==NULL)
                front =rear=new node(data);
        else
                rear->next=new node(data);
                rear=rear->next;
            }
```

c. Add Function:

Steps required for insertion of element x in queue:

1) Memory is acquired for the new node.
2) Value x is stored in the new node.
3) New node is inserted at the rear end.
4) Special care should be taken for insertion into an empty queue. Both rear and front pointers will point to the only element of the queue.

```
Void que::add(int data)
{
        if(front==NULL)
                Front=rear=new node(data);
        Else{
                        Rear->next=new node(data);
                        Rear=rear->next;
}               }
```

d. Deletion Function:

```
if(front==NULL)
    {
        cout<<"\nQueue is empty";
        return;
    }
  p=front;
  front=front->next;
  cout<<"\nDeleted job="<<p->data;
  if(front==NULL)
  rear=NULL;
 delete p;
```

e. Print Function:

```
if(front==NULL)
{
        cout<<"\nQueue is empty";
      return;
}
Cout<<"\nJobs in queue are:";
for(p=front;p!=NULL;p=p->next)
        cout<<""<<p->data;
```

Algorithm:

Step 1.Start

Step 2. Declare integers i,n,ch,data,arr[MAX],data=10,data1.

Step 3. Declare character ch.

Step 4.Declare nodes *p,*next.

Step 5.Call function create

Step 6.Call function add

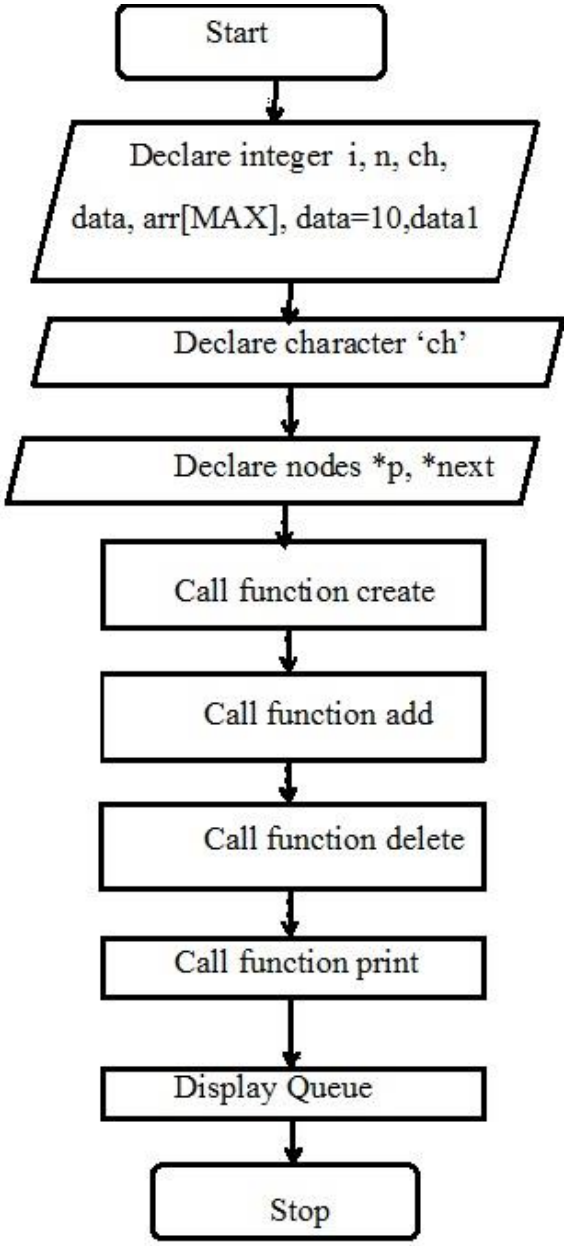
Step 7.Call function delete

Step 8.Call function print

Step 9.Display queue

Step 10.Stop

Flowchart:



Conclusion:

Here, with the help of queue using array and queue using linked list, we successfully simulate a job queue.