**ASSIGNMENT NO.-10**

**TITLE: QUEUE**

**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.

**LEARNING OBJECTIVE:**

1. Implementation of Queue.
2. Use of Queue in simulation, job scheduling and queue application.

**PRE-REQUISITES:** Operations of array and operation of queue.

**THEORY:**

* QUEUE

A queue is a special type of data structure that performs insertion at one end called rear and deletion at other end called front. The functions of queue is FIFO or LIFO. Consider an ordered list-L={a1,a2,a3...an}. Here a1 is the front and an is the rear element.

* APPLICATIONS OF QUEUE:

In operating system with network database implementation and also use in time sharing distributed computer system where user needs to share system simultaneously.

* OPERATIONS ON QUEUE:-

1. Create:

Creates an empty queue:Q.

1. Add(i,Q):

Adds an element i to the rear of queue. Q returns new queue.

1. Delete(Q):

Takes out an element from front end and returns the resulting queue.

1. Getfront(Q):

Returns the element at front position of the queue.

1. Is\_empty(Q):

Returns true if queue is empty otherwise false.

* QUEUE USING ARRAY:-

1. Create

#define MAX 50

int queue[MAX];

int front=rear=-1;

Here MAX is the maximum initial size that is defined. For 1D queue we need two variables front and rear set to -1 or 0 to represent an empty queue.

1. Is\_empty()

Bool Is\_empty

{

If (front==rear)

Return 1;

Else

Return 0;

}

1. Is\_full()

Bool Is\_full()

{

If(rear==MAX)

Return 1;

Else

Return 0;

}

1. Add

This operation adds an element as rear points to last element of queue. New element is added rear+1 location.

Void add(int element)

{

If(Is\_full())

Cout<<”Error, Q is full”;

Else

Queue[++rear]=element;

}

1. Delete()

This operation deletes an element from front of queue and sets to the next element. F can be initialized one position less than actual front.

Int delete()

{

If(Is\_empty())

Cout<<”Q is empty”;

Else

Return(queue[++front]);

}

1. Getfront()

This operation returns the element at front position but unlike delete operation it does not update value of front.

Int getfront()

{

If(Is\_emoty())

Cout<<”Q is empty”;

Else

Return(queue[front 1])

}

EXAMPLE:

0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

MAX=4

Rear=front=-1

Let Q be an empty queue with front and rear =-1 and max=5.

Q.add(11)

0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| 11 |  |  |  |

Front=-1,rear=0

Q.add(12)

0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | 12 |  |  |

Front=-1,rear=1

Q.add(13)

0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | 12 | 13 |  |

Front=-1,rear=2

Q.add(14)

0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | 12 | 13 | 14 |

Front=-1,rear=3

Q.add(15)

Queue is full.