6. APPLICATION OF QUEUE ADT – FCFS SCHEDULING

Preamble

The following are some crucial applications of Queue data structure.

Task Scheduling: Queues can be used to schedule tasks based on priority or the order in which they were received.

Resource Allocation: Queues can be used to manage and allocate resources, such as printers or CPU processing time.

Batch Processing: Queues can be used to handle batch processing jobs, such as data analysis or image rendering.

Message Buffering: Queues can be used to buffer messages in communication systems, such as message queues in messaging systems or buffers in computer networks.

Event Handling: Queues can be used to handle events in event-driven systems, such as GUI applications or simulation systems.

Traffic Management: Queues can be used to manage traffic flow in transportation systems, such as airport control systems or road networks.

Operating systems: Operating systems often use queues to manage processes and resources. For example, a process scheduler might use a queue to manage the order in which processes are executed.

Network protocols: Network protocols like TCP and UDP use queues to manage packets that are transmitted over the network. Queues can help to ensure that packets are delivered in the correct order and at the appropriate rate.

Printer queues: In printing systems, queues are used to manage the order in which print jobs are processed. Jobs are added to the queue as they are submitted, and the printer processes them in the order they were received.

Web servers: Web servers use queues to manage incoming requests from clients. Requests are added to the queue as they are received, and they are processed by the server in the order they were received.

Breadth-first search algorithm: The breadth-first search algorithm uses a queue to explore nodes in a graph level-by-level. The algorithm starts at a given node, adds its neighbors to the queue, and then processes each neighbor in turn.

Steps

Step 1: Initialize the queue variables front =0 and rear = -1

Step 2: Read the queue operation type.

Step 3: Check the queue operations status.

i). If it is Insertion then do the following steps

Check rear < queue_size is true increment the rear by one and read the queue element and also display queue. otherwise display the queue is full.

Go to step 2

ii). If it is deletion then do the following steps

Check rear< front is true then display the queue is empty.

Move the elements to one step forward (i.e. move to previous index).

Decreases the rear value by one (rear=rear-1).

Display queue

Go to step2.

Implementation in C

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int n,a[10],b[10],t[10],w[10],g[10],i,m;
      float att=0,awt=0; clrscr();
      for(i=0;i<10;i++)
      {
            a[i]=0; b[i]=0; w[i]=0; g[i]=0;
      }
      printf("enter the number of process");
      scanf("%d",&n);
      printf("enter the burst times");
      for(i=0;i<n;i++)
      scanf("%d",&b[i]);
      printf("\nenter the arrival times");
      for(i=0;i<n;i++)
```

```
        scanf("%d",&a[i]);
        g[0]=0;
        for(i=0;i<10;i++)
              g[i+1]=g[i]+b[i];
        for(i=0;i<n;i++)
        {
              w[i]=g[i]-a[i];
              t[i]=g[i+1]-a[i];
              awt=awt+w[i];
              att=att+t[i];
        }
        awt =awt/n;
        att=att/n;
        printf("\n\tprocess\twaiting time\tturn arround time\n");
        for(i=0;i<n;i++)

        {
              printf("\tp%d\t\t%d\t\t%d\n",i,w[i],t[i]);
        }
        printf("the average waiting time is %f\n",awt); printf("the average
        turn around time is %f\n",att);
        getch();
}
```

Sample Input and Output

```
enter the number of process 3
enter the burst times 0.5

enter the arrival times
        process waiting time    turn arround time
        p0              0               0
        p1              0               0
        p2              0               0
the average waiting time is 0.000000
the average turn around time is 0.000000

--------------------------------
Process exited after 41.07 seconds with return value 41
Press any key to continue . . .
```