EXPERIMENT NO:-10

AIM: - Write a program to implement Disk Scheduling algorithms like FCFS, SCAN and C-SCAN.

THEORY:-

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

Some of the important terms

- **Seek Time:**Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:**

Disk Access Time is:

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

Disk Scheduling Algorithms

FCFS: FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Let us understand this with the help of an example.

C program for FCFS disk scheduling:

```
#include<stdio.h>
int main()
{          int queue[20],n,head,i,j,k,seek=0,max,diff;
           float avg;
          printf("Enter the max range of disk\n");
          scanf("%d",&max);
```

```
        printf("Enter the size of queue request\n");
        scanf("%d",&n);
        printf("Enter the queue of disk positions to be read\n");
        for(i=1;i<=n;i++)
        scanf("%d",&queue[i]);
        printf("Enter the initial head position\n");
        scanf("%d",&head);
        queue[0]=head;
        for(j=0;j<=n-1;j++)
        {       diff=abs(queue[j+1]-queue[j]);seek+=diff;
                printf("Disk head moves from %d to %d with seek %d\n",queue[j],queue[j+1],diff);
        }
        printf("Total seek time is %d\n",seek);
        avg=seek/(float)n;
        printf("Average seek time is %f\n",avg);
        return 0;
}
```

OUTPUT :

```
user@user-H81M-S:~$ ./Exp10_1
Enter the max range of disk:
300
Enter the size of queue request:
5
Enter the queue of disk positions to be read:
90
52
176
250
12
Enter the initial head position:
100
Disk head moves from 100 to 90 with seek 10
Disk head moves from 90 to 52 with seek 38
Disk head moves from 52 to 176 with seek 124
Disk head moves from 176 to 250 with seek 74
Disk head moves from 250 to 12 with seek 238
Total seek time is 484
Average seek time is 96.800003
```

SCAN disk scheduling :

In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end,servicing requests as it reaches each cylinder,until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk. The SCAN algorithm is sometimes called the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

C program for SCAN disk scheduling :

```
#include<stdio.h>
int main()
{
        int queue[20],n,head,i,j,k,seek=0,max,diff,temp,queue1[20],queue2[20], temp1=0,temp2=0;
        float avg;
        printf("Enter the max range of disk\n");
        scanf("%d",&max);
        printf("Enter the initial head position\n");
        scanf("%d",&head);
        printf("Enter the size of queue request\n");
        scanf("%d",&n);
        printf("Enter the queue of disk positions to be read\n");
        for(i=1;i<=n;i++)
        {                   scanf("%d",&temp);
                            if(temp>=head){
                           queue1[temp1]=temp;
                           temp1++;}
                else {
                     queue2[temp2]=temp;
                     temp2++;
                     }
        }
        for(i=0;i<temp1-1;i++)
        {          for(j=i+1;j<temp1;j++)
                {
                           if(queue1[i]>queue1[j])
                           {
                                    temp=queue1[i];
                                    queue1[i]=queue1[j];
                                    queue1[j]=temp;
                           }
                }
        }
        for(i=0;i<temp2-1;i++)
        {           for(j=i+1;j<temp2;j++)
                {          if(queue2[i]<queue2[j])
                           {
                                    temp=queue2[i];
                                    queue2[i]=queue2[j];
                                    queue2[j]=temp;
                           }
                }
        }
         for(i=1,j=0;j<temp1;i++,j++)
```

```
        queue[i]=queue1[j];
        queue[i]=max;
        for(i=temp1+2,j=0;j<temp2;i++,j++)
        queue[i]=queue2[j];
        queue[i]=0;
        queue[0]=head;
        for(j=0;j<=n+1;j++)
        {       diff=abs(queue[j+1]-queue[j]);
                seek+=diff;
                 printf("Disk head moves from %d to %d with seek %d\n",queue[j],queue[j+1],diff);
        }
        printf("Total seek time is %d\n",seek);
        avg=seek/(float)n;
        printf("Average seek time is %f\n",avg);
        return 0;
}
```

OUTPUT :

```
user@user-H81M-S:~$ ./exp10_2
Enter the max range of disk:
200
Enter the initial head position:
100
Enter the size of queue request:
5
Enter the queue of disk positions to be read:
110
90
136
66
150
Disk head moves from 100 to 110 with seek 10
Disk head moves from 110 to 136 with seek 26
Disk head moves from 136 to 150 with seek 14
Disk head moves from 150 to 200 with seek 50
Disk head moves from 200 to 90 with seek 110
Disk head moves from 90 to 66 with seek 24
Disk head moves from 66 to 0 with seek 66
Total seek time is 300
Average seek time is 60.000000
```

C-SCAN disk scheduling :

Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip (Figure 10.7). The C-SCAN scheduling algorithm essentially treats the cylinders as a circularlist that wraps around from the final cylinder to the first one.

C program for C-SCAN disk scheduling :

```
#include<stdio.h>
int main()
{       int queue[20],n,head,i,j,k,seek=0,max,diff,temp,queue1[20],queue2[20], temp1=0,temp2=0;
        float avg;
        printf("Enter the max range of disk\n");
        scanf("%d",&max);
        printf("Enter the initial head position\n");
        scanf("%d",&head);
        printf("Enter the size of queue request\n");
        scanf("%d",&n);
        printf("Enter the queue of disk positions to be read\n");
        for(i=1;i<=n;i++)
        {       scanf("%d",&temp);
                if(temp>=head)
                {
                        queue1[temp1]=temp;
                        temp1++;
                }
                else
                {
                        queue2[temp2]=temp;
                        temp2++;
                }
        }
        for(i=0;i<temp1-1;i++)
        {       for(j=i+1;j<temp1;j++)
                {       if(queue1[i]>queue1[j])
                        {       temp=queue1[i];
                                queue1[i]=queue1[j];
                                queue1[j]=temp;
                        }
                }
        }
        for(i=0;i<temp2-1;i++)
        {       for(j=i+1;j<temp2;j++)
                {
                        if(queue2[i]>queue2[j])
                        {
                                temp=queue2[i];
                                queue2[i]=queue2[j];
                                queue2[j]=temp;
                        }
                }
        }
         for(i=1,j=0;j<temp1;i++,j++)
         queue[i]=queue1[j];
        queue[i]=max;
        queue[i+1]=0;
```

```
        for(i=temp1+3,j=0;j<temp2;i++,j++)
        queue[i]=queue2[j];
        queue[0]=head;
        for(j=0;j<=n+1;j++)
        {          diff=abs(queue[j+1]-queue[j]);seek+=diff;
                printf("Disk head moves from %d to %d with seek %d\n",queue[j],queue[j+1],diff);
        }
        printf("Total seek time is %d\n",seek);
        avg=seek/(float)n;
        printf("Average seek time is %f\n",avg);
        return 0;
}
```

OUTPUT :

```
user@user-H81M-S:~$ ./exp10_3_1
Enter the max range of disk:
200
Enter the initial head position:
100
Enter the size of queue request:
5
Enter the queue of disk positions to be read:
198
54
11
98
176
Disk head moves from 100 to 176 with seek 76
Disk head moves from 176 to 198 with seek 22
Disk head moves from 198 to 200 with seek 2
Disk head moves from 200 to 0 with seek 200
Disk head moves from 0 to 11 with seek 11
Disk head moves from 11 to 54 with seek 43
Disk head moves from 54 to 98 with seek 44
Total seek time is 398
Average seek time is 79.599998
```

CONCLUSION: -Thus we have studied and implemented Disk Scheduling algorithms like FCFS, SCAN and C-SCAN.

SIGN AND REMARK

R1 (3 Marks)	R2 (3 Marks)	R3 (3 Marks)	R4 (3 Mark)	R5 (3 Mark)	Total (15 Marks)	Signature