Ex.No.:10   **DISK SCHEDULING**

Date:

Aim:

To write a C program for implementation of Disk Scheduling.

FCFS:

Algorithm:

Step1: Read the current position of disk head and number of requests.

Step2: Read all requests in order.

Step3: Calculate head movement by taking absolute difference between current position and first request and add them up.

Step4: Print the path of disk head movement by taking absolute difference between adjacent elements of request array and add them up.

Step5: Print total head movement.

Code:

```
#include<math.h>

#include<stdio.h>

#include<stdlib.h>

int main()

{

    int i,n,req[50],mov=0,cp;

    printf("enter the current position\n");

    scanf("%d",&cp);

    printf("enter the number of requests\n");

    scanf("%d",&n);

    printf("enter the request order\n");

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

    {

        scanf("%d",&req[i]);

    }

    mov=mov+abs(cp-req[0]); // abs is used to calculate the absolute value
```

```c
    printf("%d -> %d",cp,req[0]);

    for(i=1;i<n;i++)

    {

        mov=mov+abs(req[i]-req[i-1]);

        printf(" -> %d",req[i]);

    }

    printf("\n");

    printf("total head movement = %d\n",mov);

    getch();

    return 0;

}
```

Output:

SSTF:

Algorithm:

Step1: Read the current position, number of requests and request order.

Step2: Calculate distance of each request from current position.

Step3: Find the nearest request by finding the minimum distance.

Step4: Add the nearest request to a new array and change the current position value to next request.

Step5: Calculate head movement by taking absolute difference between adjacent elements of new array and add them up.

Code:

```c
#include<math.h>
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,n,k,req[50],mov=0,cp,index[50],min,a[50],j=0,mini,cp1;
    printf("enter the current position\n");
    scanf("%d",&cp);
    printf("enter the number of requests\n");
    scanf("%d",&n);
    cp1=cp;
    printf("enter the request order\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&req[i]);
    }
    for(k=0;k<n;k++)
    {
    for(i=0;i<n;i++)
    {
        index[i]=abs(cp-req[i]); // calculate distance of each request from current position
    }
    // to find the nearest request
    min=index[0];
    mini=0;
    for(i=1;i<n;i++)
    {
        if(min>index[i])
```

```c
    {
    min=index[i];mini=i;
         }
       }
     a[j]=req[mini];
     j++;
     cp=req[mini]; // change the current position value to next request
     req[mini]=999;
     } // the request that is processed its value is changed so that it is not processed again
     printf("Sequence is : ");
     printf("%d",cp1);
     mov=mov+abs(cp1-a[0]);   // head movement
     printf(" -> %d",a[0]);
     for(i=1;i<n;i++)
     {
       mov=mov+abs(a[i]-a[i-1]); ///head movement
       printf(" -> %d",a[i]);
     }
     printf("\n");
     printf("total head movement = %d\n",mov);
     getch();
     return 0;
```

Output:

```
C:\Users\21cse\OneDrive\Doc   ×   +   ∨
enter the current position
60
enter the number of requests
6
enter the request order
34
404
43
23
53
45
Sequence is : 60 -> 53 -> 45 -> 43 -> 34 -> 23 -> 404
total head movement = 418

--------------------------------
Process exited after 33.92 seconds with return value 0
Press any key to continue . . .
```

SCAN:

Algorithm:

Step1: Read the maximum range of disk, initial head position, size of queue request and queue of disk positions to be read.

Step2: Divide the queue into two parts: one part containing all the elements greater than or equal to head and other part containing all the elements less than head.

Step3: Sort both parts of the queue in ascending and descending order respectively.

Step4: Merge both parts of the queue in a single queue in order of their execution.

Step5: Calculate seek time by taking absolute difference between adjacent elements of queue and add them up.

Code:

```c
#include <stdio.h>

#include <math.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);
```

```c
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;

        }
```

```c
        }
    }
    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);
    }
```

```c
    printf("Total seek time is %d\n", seek);

    avg = seek / (float)n;

    printf("Average seek time is %f\n", avg);

    getch();

    return 0;

}
```

Output:

C-SCAN:

Algorithm:

Step1: Read the max range of disk

Step2: Read initial head position

Step3: Read size of queue request

Step4: Read queue of disk positions to be read

Step5: Calculate total seek time and average seek time

Code:

```c
#include <stdio.h>

#include <math.h>

int main()

{

    int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],
```

```c
    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])
            {
```

```c
            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;
```
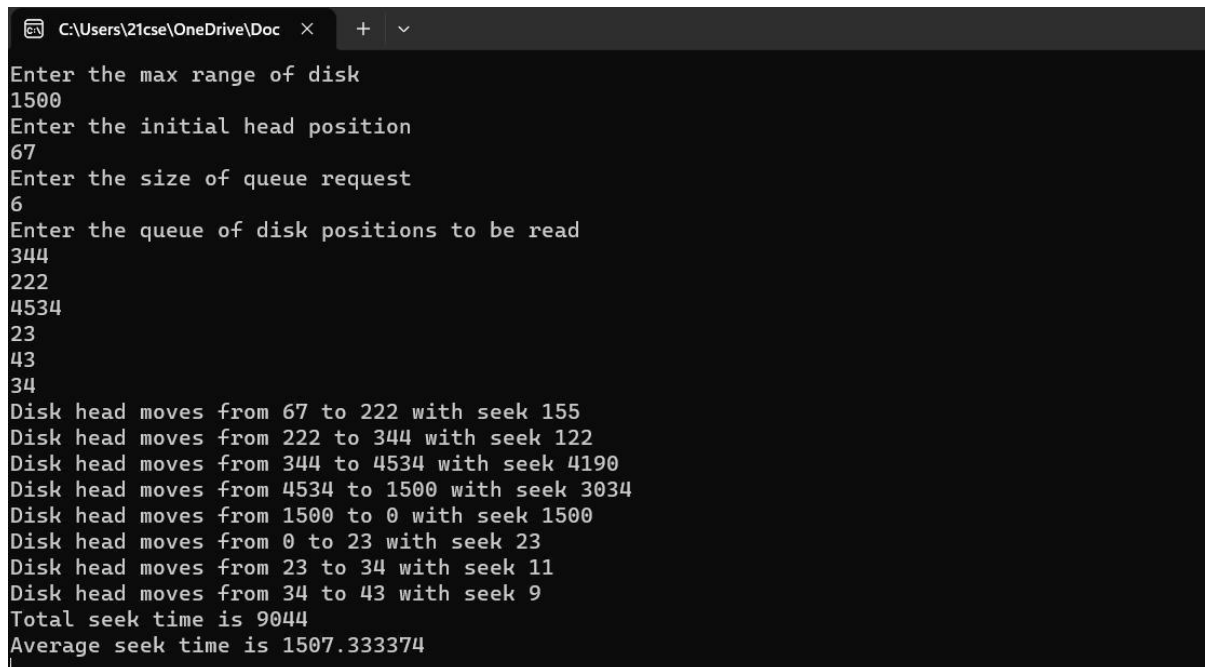
```
    printf("Disk head moves from %d to %d with seek %d\n", queue[j],

    queue[j + 1], diff);

    }

    printf("Total seek time is %d\n", seek);

    avg = seek / (float)n;

    printf("Average seek time is %f\n", avg);

    getch();

    return 0;

}
```

Output:

```
 C:\Users\21cse\OneDrive\Doc  ×    +  ∨
Enter the max range of disk
1500
Enter the initial head position
67
Enter the size of queue request
6
Enter the queue of disk positions to be read
344
222
4534
23
43
34
Disk head moves from 67 to 222 with seek 155
Disk head moves from 222 to 344 with seek 122
Disk head moves from 344 to 4534 with seek 4190
Disk head moves from 4534 to 1500 with seek 3034
Disk head moves from 1500 to 0 with seek 1500
Disk head moves from 0 to 23 with seek 23
Disk head moves from 23 to 34 with seek 11
Disk head moves from 34 to 43 with seek 9
Total seek time is 9044
Average seek time is 1507.333374
```

LOOK:

Algorithm:

Step1: Read the total number of requests and their sequence.

Step2: Read the initial head position and total disk size.

Step3: Read the head movement direction (1 for high and 0 for low).

Step4: Sort the requests in ascending order.

Step5: Calculate the total head movement by traversing the requests in two parts:

Step6: If movement is towards high value:

Step7: Traverse from initial position to the highest request index.

Step8: Traverse from lowest request index to initial position.

Step9: If movement is towards low value:

Step10: Traverse from initial position to lowest request index.

Step11: Traverse from highest request index to initial position.

Code:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
    printf("Enter the number of Requests\n");
    scanf("%d",&n);
    printf("Enter the Requests sequence\n");
    for(i=0;i<n;i++)
     scanf("%d",&RQ[i]);
    printf("Enter initial head position\n");
    scanf("%d",&initial);
    printf("Enter total disk size\n");
    scanf("%d",&size);
    printf("Enter the head movement direction for high 1 and for low 0\n");
    scanf("%d",&move);
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-i-1;j++)
```

```
   {
     if(RQ[j]>RQ[j+1])
     {
        int temp;
        temp=RQ[j];
        RQ[j]=RQ[j+1];
        RQ[j+1]=temp;
     }
   }
}
int index;
for(i=0;i<n;i++)
{
   if(initial<RQ[i])
   {
      index=i;
      break;
   }
}
// if movement is towards high value
if(move==1)
{
   for(i=index;i<n;i++)
   {
     TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
     initial=RQ[i];
   }
   for(i=index-1;i>=0;i--)
   {
```

```c
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

            initial=RQ[i];

        }

    }

    // if movement is towards low value

    else

    {

        for(i=index-1;i>=0;i--)

        {

            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

            initial=RQ[i];

        }

        for(i=index;i<n;i++)

        {

            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

            initial=RQ[i];

        }

    }

    printf("Total head movement is %d",TotalHeadMoment);

    return 0;

}
```

Output:

```
C:\Users\21cse\OneDrive\Doc   ×   +   ∨

Enter the number of Requests
6
Enter the Requests sequence
34
366
78
90
644
345
Enter initial head position
64
Enter total disk size
2000
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 1190
--------------------------------
Process exited after 19.81 seconds with return value 0
Press any key to continue . . .
```

C-LOOK:

Algorithm:

Step1: Read the maximum range of disk, number of queue requests and initial head position.

Step2: Divide the queue into two parts: one with requests greater than head position and other with requests less than or equal to head position.

Step3: Sort both parts in ascending order.

Step4: Add all requests greater than head position to a new array in ascending order followed by head position and then all requests less than or equal to head position in ascending order.

Step5: Calculate head movement by taking absolute difference between adjacent elements of new array and add them up.

Code:

```
#include<stdio.h>

int absoluteValue(int);

void main()

{

    int queue[25],n,headposition,i,j,k,seek=0, maxrange,

    difference,temp,queue1[20],queue2[20],temp1=0,temp2=0;

    float averageSeekTime;
```

```c
printf("Enter the maximum range of Disk: ");
scanf("%d",&maxrange);
printf("Enter the number of queue requests: ");
scanf("%d",&n);
printf("Enter the initial head position: ");
scanf("%d",&headposition);
printf("Enter the disk positions to be read(queue): ");
for(i=1;i<=n;i++)
{
    scanf("%d",&temp);
    if(temp>headposition)
    {
        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]=queue2[0];

    for(i=temp1+1,j=0;j<temp2;i++,j++)

    {

        queue[i]=queue2[j];

    }

    queue[0]=headposition;

    for(j=0; j<n; j++)

    {

        difference = absoluteValue(queue[j+1]-queue[j]);
```

```
        seek = seek + difference;

        printf("Disk head moves from position %d to %d with Seek %d \n",

        queue[j], queue[j+1], difference);

    }

    averageSeekTime = seek/(float)n;

    printf("Total Seek Time= %d\n", seek);

    printf("Average Seek Time= %f\n", averageSeekTime);

}

int absoluteValue(int x)

{

    if(x>0)

    else

    {

        return x*-1;

    }

}
```

Output:


```
C:\Users\21cse\OneDrive\Doc    ×    +    ∨
Enter the maximum range of Disk: 2000
Enter the number of queue requests: 5
Enter the initial head position: 78
Enter the disk positions to be read(queue): 45
34
56
4778
34
Disk head moves from position 78 to 4778 with Seek 4700
Disk head moves from position 4778 to 34 with Seek 4744
Disk head moves from position 34 to 34 with Seek 0
Disk head moves from position 34 to 45 with Seek 11
Disk head moves from position 45 to 56 with Seek 11
Total Seek Time= 9466
Average Seek Time= 1893.199951

--------------------------------
Process exited after 14.81 seconds with return value 31
Press any key to continue . . .
```

Observation	
Record	
Total	
Initial	

Result:

      Thus the C program for implementation of Disk Scheduling are written and executed successfully.

Name:B.Ajeetha   Roll No:21CSE067