**Name : Kunal Sachin Kharat**

**Roll no : 33**

**Class : CSAIML-A**

**Q.) Implement Disk Scheduling Algorithms:**

1. **FCFS(First Come First Serve):**

#include<stdio.h>

#include<stdlib.h>

int main()

{

    int total\_head\_movement=0,initial\_pos,n ;

    printf("\n Enter the no. of cylinders in Request queue:\n");

    scanf("%d",&n);

    int request\_queue[n];

    printf("\n Enter the cylinders no. in Request queue :\n");

    for(int i=0;i<n;i++)

     scanf("%d",&request\_queue[i]);

    printf("\n Enter the initial Position of RW head: ");

    scanf("%d",&initial\_pos);

    for(int i=0;i<n;i++)

    {

        total\_head\_movement += abs(initial\_pos - request\_queue[i]);

        initial\_pos = request\_queue[i];

    }

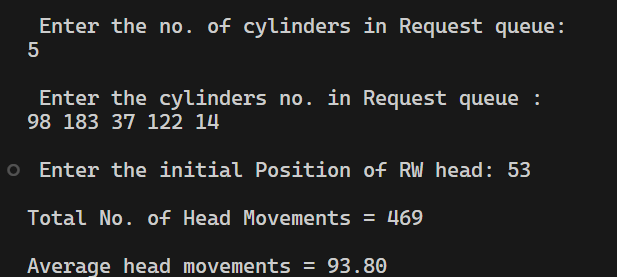
   printf("\nTotal No. of Head Movements = %d\n",total\_head\_movement);

   printf("\nAverage head movements = %.2f\n",(float)total\_head\_movement/n);

   return 0;

}

**Output:**



1. **SSTF:**

/\*

  SSTF Disk Scheduling Algorithm

\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main() {

  int queue[100], queue2[100], q\_size, head, seek=0, temp;

  float avg;

  printf("%s\n", "-----SSTF Disk Scheduling Algorithm-----");

  printf("%s\n", "Enter the size of the queue");

  scanf("%d", &q\_size);

  printf("%s\n", "Enter queue elements");

  for(int i=0; i<q\_size; i++){

    scanf("%d",&queue[i]);

  }

  printf("%s\n","Enter initial head position");

  scanf("%d", &head);

  //get distance from head of elems in queue

  for(int i=0; i<q\_size; i++){

    queue2[i] = abs(head-queue[i]);

  }

  //swap elems based on their distance from each other

  for(int i=0; i<q\_size; i++){

      for(int j=i+1; j<q\_size;j++){

        if(queue2[i]>queue2[j]){

            temp = queue2[i];

            queue2[i]=queue[j];

            queue2[j]=temp;

            temp=queue[i];

            queue[i]=queue[j];

            queue[j]=temp;

        }

    }

  }

  for(int i=1; i<q\_size; i++){

    seek = seek+abs(head-queue[i]);

    head = queue[i];

  }

  printf("\nTotal seek time is %d\t",seek);

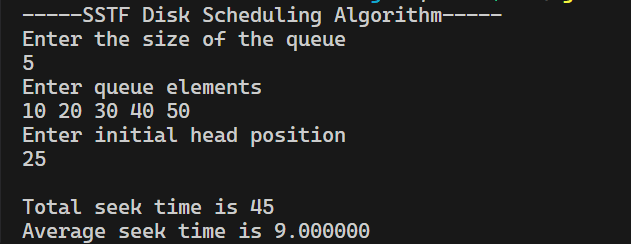
  avg = seek/(float)q\_size;

  printf("\nAverage seek time is %f\t", avg);

  return 0;

}

**Output:**



1. **SCAN:**

// C program to implement SCAN Disk Scheduling algorithm

#include<stdio.h>

#include<stdlib.h>

#include<limits.h>

int comparator(const void \* a, const void \*b)

{

   int x =\*(int \*)a;

   int y =\*(int \*)b;

   if(x<y)

     return -1;  // No sorting

   else if( x>=y) // = is for stable sort

    return 1;    // Sort

}

int min\_element(int request\_queue[],int n)

{

    int min = INT\_MAX;

    for(int i=0;i<n;i++)

    {

       if(request\_queue[i] < min)

          min = request\_queue[i];

    }

    return min;

}

int max\_element(int request\_queue[],int n)

{

   int max = INT\_MIN;

   for(int i=0;i<n;i++)

   {

      if(request\_queue[i] > max)

          max = request\_queue[i];

   }

    return max;

}

int applySCANAlgo(int total\_cylinders,int request\_queue[], int initial\_pos, int seek\_sequence[], int direction,int \*sequence\_size,int n)

{

    int total\_head\_movement=0,j=0,k=0;

    int left[n+1], right[n+1];  //in worst case(corner cases), size will be n+1

    // appending end values

    // which has to be visited

    // before reversing the direction

    if(direction == 0)

    {

        if(initial\_pos > min\_element(request\_queue,n))

            right[j++]=total\_cylinders-1;

        right[j++]=initial\_pos;

    }

    else if(direction == 1)

    {

        if(initial\_pos < max\_element(request\_queue,n))

           left[k++]=0;  //here  0 is initial cylinder of HDD

        left[k++]=initial\_pos;

    }

    for (int i = 0; i<n; i++) {

        if (request\_queue[i] < initial\_pos)

            left[k++]=request\_queue[i];

        if (request\_queue[i] > initial\_pos)

            right[j++]=request\_queue[i];

    }

    // sorting left and right arrays

    qsort((void \*)left,k, sizeof(int),comparator);

    qsort((void \*)right,j, sizeof(int),comparator);

    // run the while loop two times one by one scanning right and left of the head

    int completed = 2;

    while (completed--)

    {

        if (direction == 0)

        {

            for (int i = 0; i < j; i++)

            {

                // calculate absolute distance

                total\_head\_movement += abs(initial\_pos - right[i]);

                initial\_pos = right[i];

                // appending current track to seek sequence

                seek\_sequence[\*sequence\_size]=right[i];

                (\*sequence\_size)++;

            }

            direction = 1;

        }

        else if (direction == 1)

        {

            for (int i = k - 1; i >= 0; i--)

            {

                // calculate absolute distance

                total\_head\_movement +=  abs(initial\_pos - left[i]);

                initial\_pos = left[i];

                // appending current track to seek sequence

                seek\_sequence[\*sequence\_size]=left[i];

                (\*sequence\_size)++;

            }

            direction = 0;

        }

    }  //end of while

    return total\_head\_movement;

}  //end of applySCANALgo()

int main()

{

    int total\_cylinders,total\_head\_movement=0,initial\_pos,n,direction,pos;

    printf("\nEnter the total no. of cylinders in HDD:\n");

    scanf("%d",&total\_cylinders);

    printf("\nEnter the no. of cylinders to enter in Request queue:\n");

    scanf("%d",&n);

    int request\_queue[n];

    int seek\_sequence[n+10];   // i.e take somewaht bigger size than n

    int sequence\_size=0;

    printf("\nEnter the cylinders no. in Request queue :\n");

    for(int i=0;i<n;i++)

     scanf("%d",&request\_queue[i]);

    printf("\nEnter the initial Position of RW head: ");

    scanf("%d",&initial\_pos);

    printf("\nEnter the direction in which Read Write head is moving:\n ");

    printf("\nEnter 0 if moving to higher cylinder else Enter 1: ");

    scanf("%d",&direction);

    if(initial\_pos < 0 || initial\_pos > total\_cylinders - 1)

    {

        printf("Wrong Initial Position Enetered !!");

        exit(0);

    }

    total\_head\_movement = applySCANAlgo(total\_cylinders,request\_queue,initial\_pos,seek\_sequence,direction,&sequence\_size,n);

    // \*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*

    printf("\n\n\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*");

    printf("\nSeek Sequence: ");

    for(int i=0;i<sequence\_size;i++)

     printf("%d ",seek\_sequence[i]);

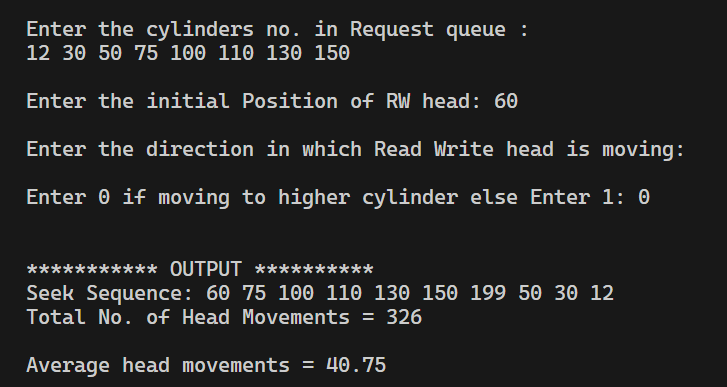
    printf("\nTotal No. of Head Movements = %d\n",total\_head\_movement);

    printf("\nAverage head movements = %.2f\n\n",(float)total\_head\_movement/n);

    return 0;

}

**Output:**



1. **C-SCAN:**

// C program to implement CSCAN Disk Scheduling algorithm

#include<stdio.h>

#include<stdlib.h>

#include<limits.h>

int comparator(const void \* a, const void \*b)

{

   int x =\*(int \*)a;

   int y =\*(int \*)b;

   if(x<y)

     return -1;  // No sorting

   else if( x>=y) // = is for stable sort

    return 1;    // Sort

}

int min\_element(int request\_queue[],int n)

{

    int min = INT\_MAX;

    for(int i=0;i<n;i++)

    {

       if(request\_queue[i] < min)

          min = request\_queue[i];

    }

    return min;

}

int max\_element(int request\_queue[],int n)

{

   int max = INT\_MIN;

   for(int i=0;i<n;i++)

   {

      if(request\_queue[i] > max)

          max = request\_queue[i];

   }

    return max;

}

int moveRight(int left[],int right[],int j,int k,int total\_cylinders, int request\_queue[], int initial\_pos, int seek\_sequence[],int \*sequence\_size,int n)

{

    //j is right array size and k is left array size

   int total\_head\_movement=0;

   for (int i = 0; i < j; i++)

   {

        // calculate absolute distance

        total\_head\_movement += abs(initial\_pos - right[i]);

        initial\_pos = right[i];

        // appending current track to seek sequence

        seek\_sequence[\*sequence\_size]=right[i];

        (\*sequence\_size)++;

   }

   k ?  total\_head\_movement += (total\_cylinders - 1): 0 ;

   initial\_pos = 0;

   //move right again if needed (i.e if left array is not empty)

   for (int i = 0; i < k; i++)

   {

        // calculate absolute distance

        total\_head\_movement += abs(initial\_pos - left[i]);

        initial\_pos = left[i];

        // appending current track to seek sequence

        seek\_sequence[\*sequence\_size]=left[i];

        (\*sequence\_size)++;

   }

   return total\_head\_movement;

}

int moveLeft(int left[],int right[],int j, int k,int total\_cylinders, int request\_queue[], int initial\_pos, int seek\_sequence[],int \*sequence\_size,int n)

{

   int total\_head\_movement=0;

   for (int i = k - 1; i >=0; i--)

   {

        // calculate absolute distance

        total\_head\_movement += abs(initial\_pos - left[i]);

        initial\_pos = left[i];

        // appending current track to seek sequence

        seek\_sequence[\*sequence\_size]=left[i];

        (\*sequence\_size)++;

   }

   j ?  total\_head\_movement += (total\_cylinders - 1): 0 ;

   initial\_pos = total\_cylinders - 1;

   //move right

   for (int i = j -1; i >=0; i--)

   {

        // calculate absolute distance

        total\_head\_movement += abs(initial\_pos - right[i]);

        initial\_pos = right[i];

        // appending current track to seek sequence

        seek\_sequence[\*sequence\_size]=right[i];

        (\*sequence\_size)++;

   }

   return total\_head\_movement;

}

int applyCSCANAlgo(int total\_cylinders, int request\_queue[], int initial\_pos, int seek\_sequence[], int \* sequence\_size,int direction,int n)

{

    int total\_head\_movement=0,j=0,k=0;

    int left[n+1], right[n+1];  //in worst case(corner cases), size will be n+1

    // appending end values which has to be visited during reversing the direction

    if( ( initial\_pos > min\_element(request\_queue,n)) && (initial\_pos < max\_element(request\_queue,n)) )

    {

          right[j++] = total\_cylinders - 1;

          left[k++] = 0;  //here  0 is initial cylinder of HDD

    }

    //Decide on basis of direction where to put initial position (this is only needed if we want to print this initia\_pos also

    // in seek\_sequence otherwise remove thsi if-else

    if(direction == 0)

        right[j++] = initial\_pos;

    else if(direction == 1)

        left[k++] = initial\_pos;

    for (int i = 0; i<n; i++)

    {

        if (request\_queue[i] < initial\_pos)

            left[k++]=request\_queue[i];

        if (request\_queue[i] > initial\_pos)

            right[j++]=request\_queue[i];

    }

    // sorting left and right arrays

    qsort((void \*)left,k, sizeof(int),comparator);

    qsort((void \*)right,j, sizeof(int),comparator);

    if(direction == 0 ) //right

        total\_head\_movement += moveRight(left, right,j,k, total\_cylinders,request\_queue, initial\_pos, seek\_sequence,sequence\_size,n);

    else if(direction == 1 ) //left

       total\_head\_movement += moveLeft( left,right,j,k, total\_cylinders, request\_queue,  initial\_pos, seek\_sequence,sequence\_size,n);

    return total\_head\_movement;

}

int main()

{

    int total\_cylinders,total\_head\_movement=0,initial\_pos,n,direction,pos;

    printf("\nEnter the total no. of cylinders in HDD:\n");

    scanf("%d",&total\_cylinders);

    printf("\nEnter the no. of cylinders to enter in Request queue:\n");

    scanf("%d",&n);

    int request\_queue[n];

    int seek\_sequence[n+10];   // i.e take somewhat bigger size than n

    int sequence\_size=0;

    printf("\nEnter the cylinders no. in Request queue :\n");

    for(int i=0;i<n;i++)

     scanf("%d",&request\_queue[i]);

    printf("\nEnter the initial Position of RW head: ");

    scanf("%d",&initial\_pos);

    printf("\nEnter the direction in which Read Write head is moving:\n ");

    printf("\nEnter 0 if moving to higher cylinder else Enter 1: ");

    scanf("%d",&direction);

    if(initial\_pos < 0 || initial\_pos > total\_cylinders - 1)

    {

        printf("Wrong Initial Position Enetered !!");

        exit(0);

    }

    total\_head\_movement = applyCSCANAlgo(total\_cylinders, request\_queue,initial\_pos,seek\_sequence,&sequence\_size,direction,n);

    // \*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*

    printf("\n\n\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*");

    printf("\nSeek Sequence: ");

    for(int i=0;i<sequence\_size;i++)

     printf("%d ",seek\_sequence[i]);

    printf("\nTotal No. of Head Movements = %d\n",total\_head\_movement);

    printf("\nAverage head movements = %.2f\n\n",(float)total\_head\_movement/n);

    return 0;

}

**Output:**

