**Lab no:13  Date:**

**Title: Write a program to simulate SCAN disk scheduling algorithm**

**SCAN (Elevator):** Also known as elevator algorithm because the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path.

**Algorithm:**

Step 1: Start

Step 2: Request array represents the sequence of requests.

Step 3: Determine the direction of head movement

Step 4: Services the request that lies in direction.

Step 5: Increment the total seek count with this distance.

Step 6: Currently serviced track position now becomes the new head position.

Step 7: Reaches to the end of the cylinder and returns back to opposite direction and services the other request.

Step 8: Stop

**Source code:**

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  int main()  {  int REQ[100],i,j,n,TotalMove=0,initial,size,move;  printf("nter the no of request\n");  scanf("%d",&n);  printf("enter the Requests sequence\n");  for(i=0;i<n;i++)  {  scanf("%d",&REQ[i]);  }  printf("Enter initial head position\n");  scanf("%d",&initial);  printf("enter the total disk size\n");  scanf("%d",&size);  printf("Enter the head movement direction 1 for higher 0 for lower\n");  scanf("%d",&move);    for(i=0;i<n;i++) *//initially do the sorting*  {  for(j=0;j<n-i-1;j++)  {  if(REQ[j]>REQ[j+1])  {  int temp;  temp=REQ[j];  REQ[j]=REQ[j+1];  REQ[j+1]=temp;  }  }  }  int index;  for(i=0;i<n;i++)  {  if(initial<REQ[i])  {  index=i;  break;  }  }    *// if direction is towards high*  if(move==1)  {  for(i=index;i<n;i++)  {  TotalMove=TotalMove+abs(REQ[i]-initial);  initial=REQ[i];  }  TotalMove=TotalMove+abs(size-REQ[i-1]-1*); //movement for max size*  initial = size-1;  for(i=index-1;i>=0;i--)  {  TotalMove=TotalMove+abs(REQ[i]-initial);  initial=REQ[i];    }  }  *// if direction is towards low*  else  {  for(i=index-1;i>=0;i--)  {  TotalMove=TotalMove+abs(REQ[i]-initial);  initial=REQ[i];  }  TotalMove=TotalMove+abs(REQ[i+1]-0*); //movement for min size*  initial =0;  for(i=index;i<n;i++)  {  TotalMove=TotalMove+abs(REQ[i]-initial);  initial=REQ[i];    }  }    printf("Total movement is %d",TotalMove);  return 0;  } |

**Output:**

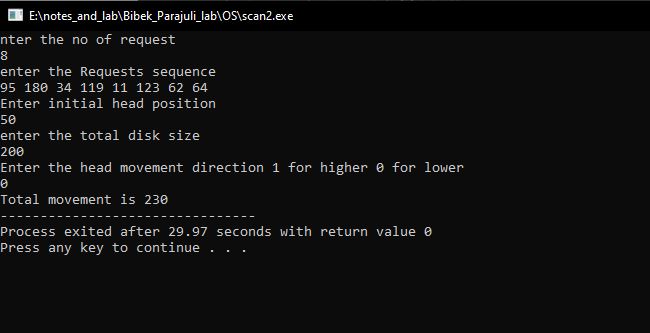


Fig: SCAN disk scheduling algorithm