**STUDENT NAME: P. HASINI**

**REG NUMBER : 192111535**

**SUBJECT NAME : OPERATING SYSTEM WITH DESIGN PRINCIPLES**

**SUBJECT CODE : CSA0470**

DAY 5 PROGRAMS(SEP 29,2022)

**21.** **FILE ALLOCATTION STRATEGY**- With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file.  Each block contains a pointer to the next block. Design a C program to simulate the file allocation strategy.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

int f[50], p,i, st, len, j, c, k, a;

for(i=0;i<50;i++)

f[i]=0;

printf("Enter how many blocks already allocated: ");

scanf("%d",&p);

printf("Enter blocks already allocated: ");

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

{

scanf("%d",&a);

f[a]=1;

}

x: printf("Enter index starting block and length: ");

scanf("%d%d", &st,&len);

k=len;

if(f[st]==0)

{

for(j=st;j<(st+k);j++)

{

if(f[j]==0)

{

f[j]=1;

printf("%d-------->%d\n",j,f[j]);

}

else

{

printf("%d Block is already allocated \n",j);

k++;

}

}

}

else

printf("%d starting block is already allocated \n",st);

printf("Do you want to enter more file(Yes - 1/No - 0)");

scanf("%d", &c);

if(c==1)

goto x;

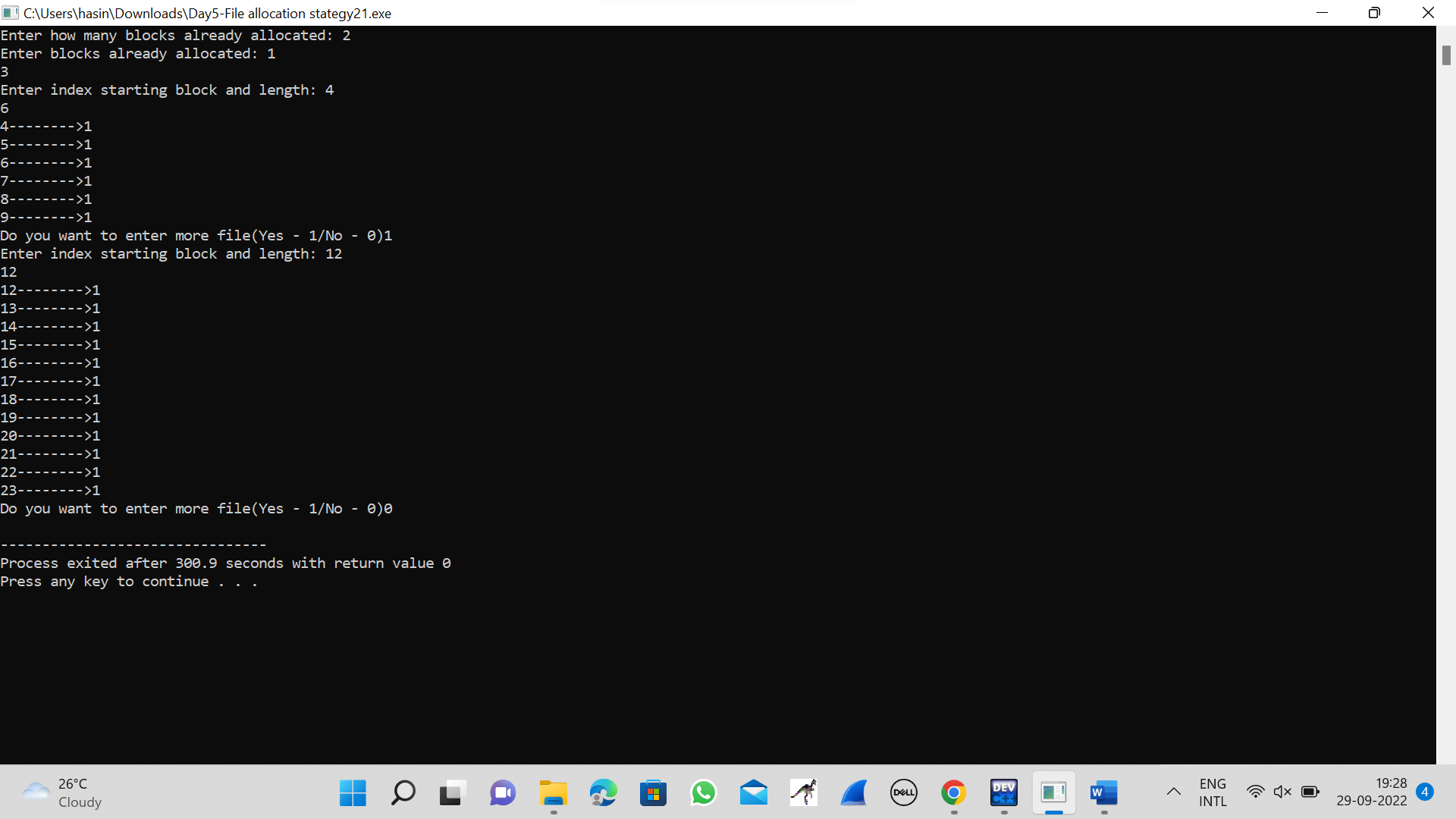
else

exit(0);

getchar();

}

**OUTPUT**

****

**22.FIRST COME FIRST SEVED DISK SCHEDULLING ALGORITHM**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int RQ[100],i,n,TotalHeadMoment=0,initial;

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);

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

{

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

initial=RQ[i];

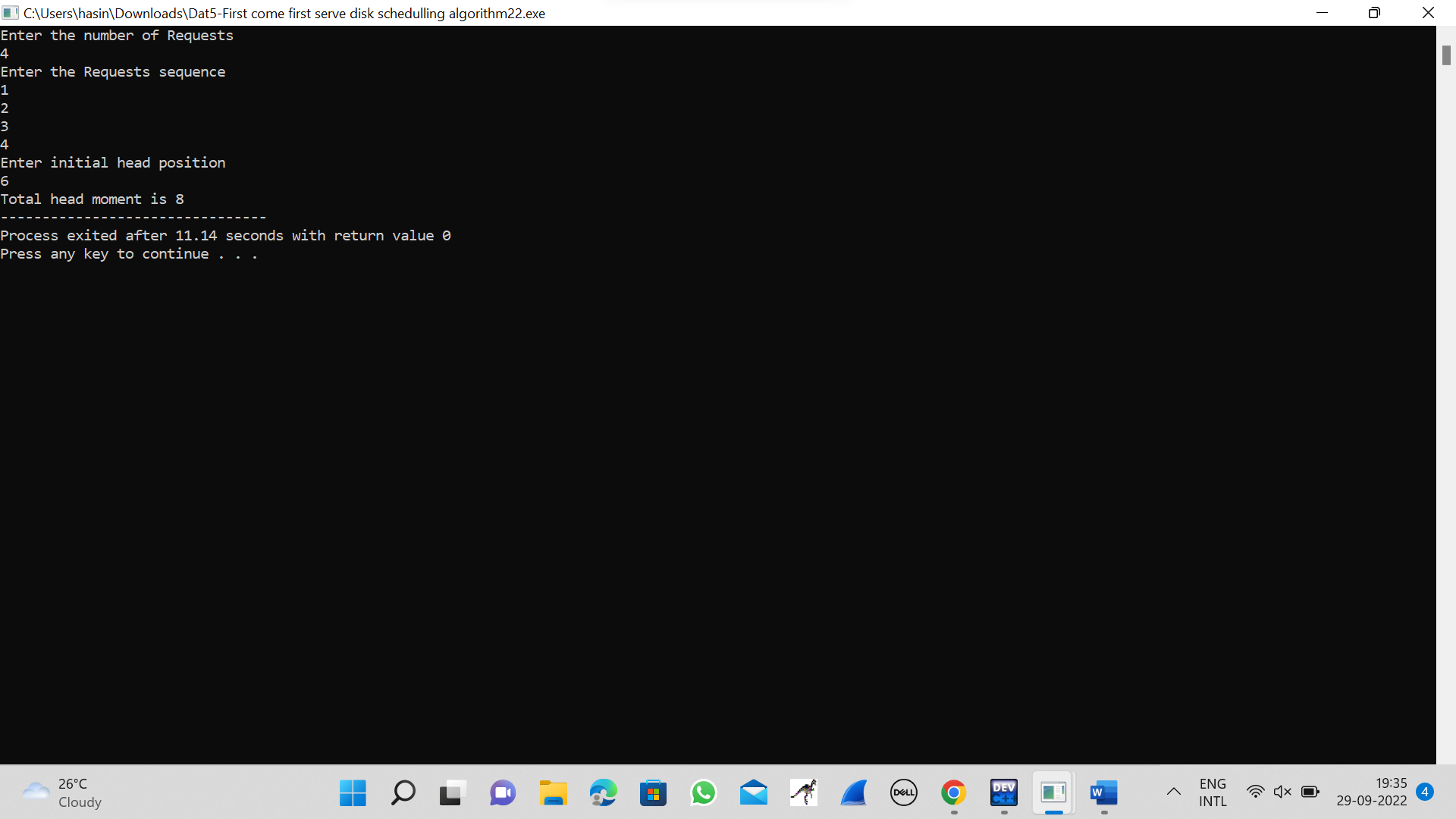
}

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

return 0;

}

**OUTPUT**

****

**23.SCAN DISK SCHEDULLING ALGORITHM**

#include<stdio.h>

int request[50];

int SIZE;

int pre;

int head;

int uptrack;

int downtrack;

struct max{

int up;

int down;

}kate[50];

int dist(int a,int b)

{

if(a>b)

return a-b;

return b-a;

}

void sort(int n)

{

int i,j;

for (i = 0; i < n-1; i++)

{

for (j = 0; j < n-i-1; j++)

{

if (request[j] > request[j+1])

{

int temp=request[j];

request[j]=request[j+1];

request[j+1]=temp;

}

}

}

j=0;

i=0;

while(request[i]!=head)

{

kate[j].down=request[i];

j++;

i++;

}

downtrack=j;

i++;

j=0;

while(i<n)

{

kate[j].up=request[i];

j++;

i++;

}

uptrack=j;

}

void scan(int n)

{

int i;

int seekcount=0;

printf("SEEK SEQUENCE = ");

sort(n);

if(pre<head){

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

{

printf("%d ",head);

seekcount=seekcount+dist(head,kate[i].up);

head=kate[i].up;

}

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

{

printf("%d ",head);

seekcount=seekcount+dist(head,kate[i].down);

head=kate[i].down;

}

}

else

{

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

{

printf("%d ",head);

seekcount=seekcount+dist(head,kate[i].down);

head=kate[i].down;

}

for(i=0;i<uptrack-1;i++)

{

printf("%d ",head);

seekcount=seekcount+dist(head,kate[i].up);

head=kate[i].up;

}

}

printf(" %d\nTOTAL DISTANCE :%d",head,seekcount);

}

int main()

{

int n,i;

printf("ENTER THE DISK SIZE :\n");

scanf("%d",&SIZE);

printf("ENTER THE NO OF REQUEST SEQUENCE :\n");

scanf("%d",&n);

printf("ENTER THE REQUEST SEQUENCE :\n");

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

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

printf("ENTER THE CURRENT HEAD :\n");

scanf("%d",&head);

request[n]=head;

request[n+1]=SIZE-1;

request[n+2]=0;

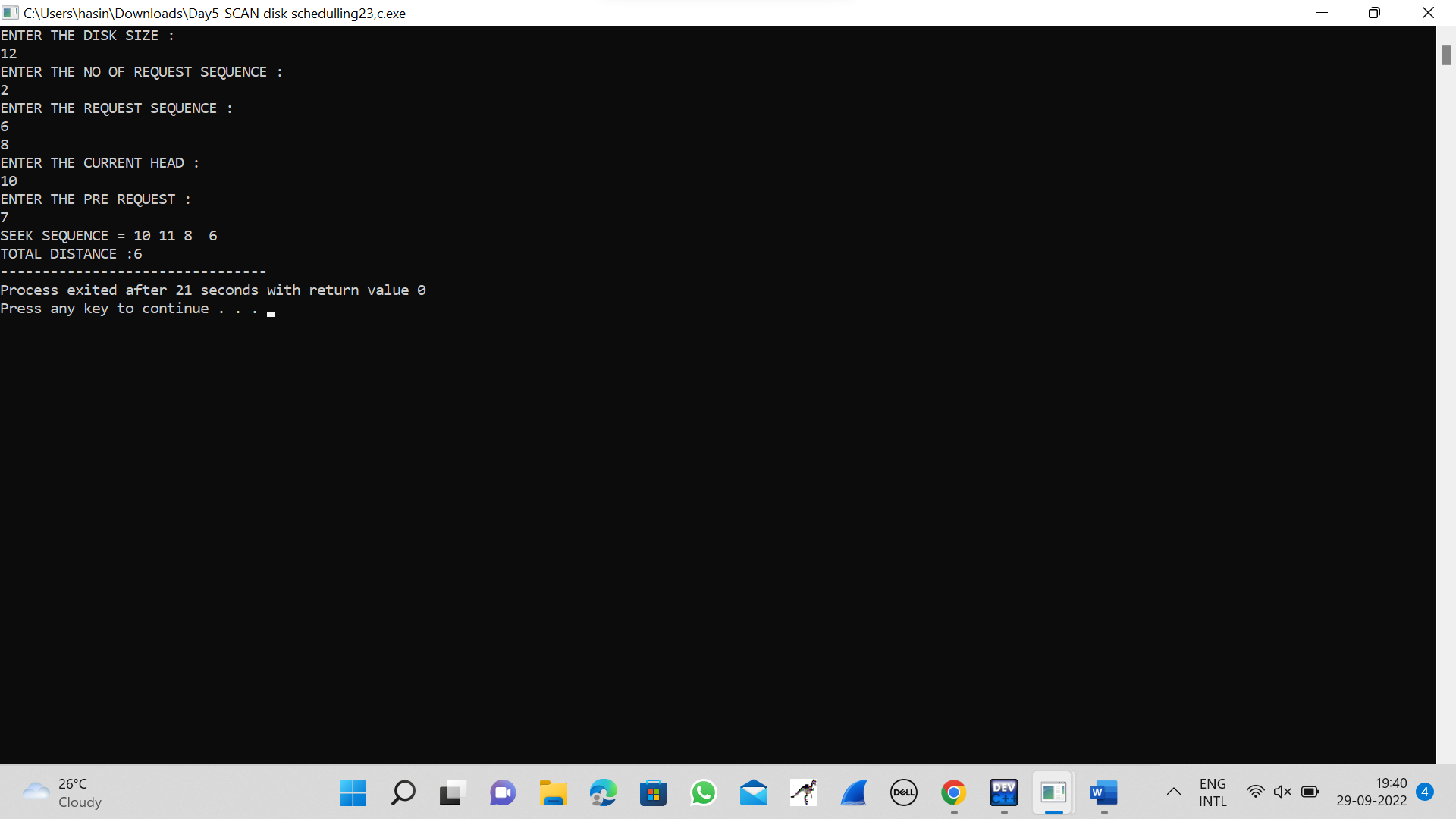
printf("ENTER THE PRE REQUEST :\n");

scanf("%d",&pre);

scan(n+3);

}

**OUTPUT**

****

**24.C-SACN DISK SCHEDULLING ALGORITHM**

#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 (move == 1){

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

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

initial = RQ[i];

}

TotalHeadMoment = TotalHeadMoment + abs(size - RQ[i - 1] - 1);

TotalHeadMoment = TotalHeadMoment + abs(size - 1 - 0);

initial = 0;

for (i = 0; i < index; i++){

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

initial = RQ[i];

}

}

else{

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

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

initial = RQ[i];

}

TotalHeadMoment = TotalHeadMoment + abs(RQ[i + 1] - 0);

TotalHeadMoment = TotalHeadMoment + abs(size - 1 - 0);

initial = size - 1;

for (i = n - 1; i >= index; i--){

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

initial = RQ[i];

}

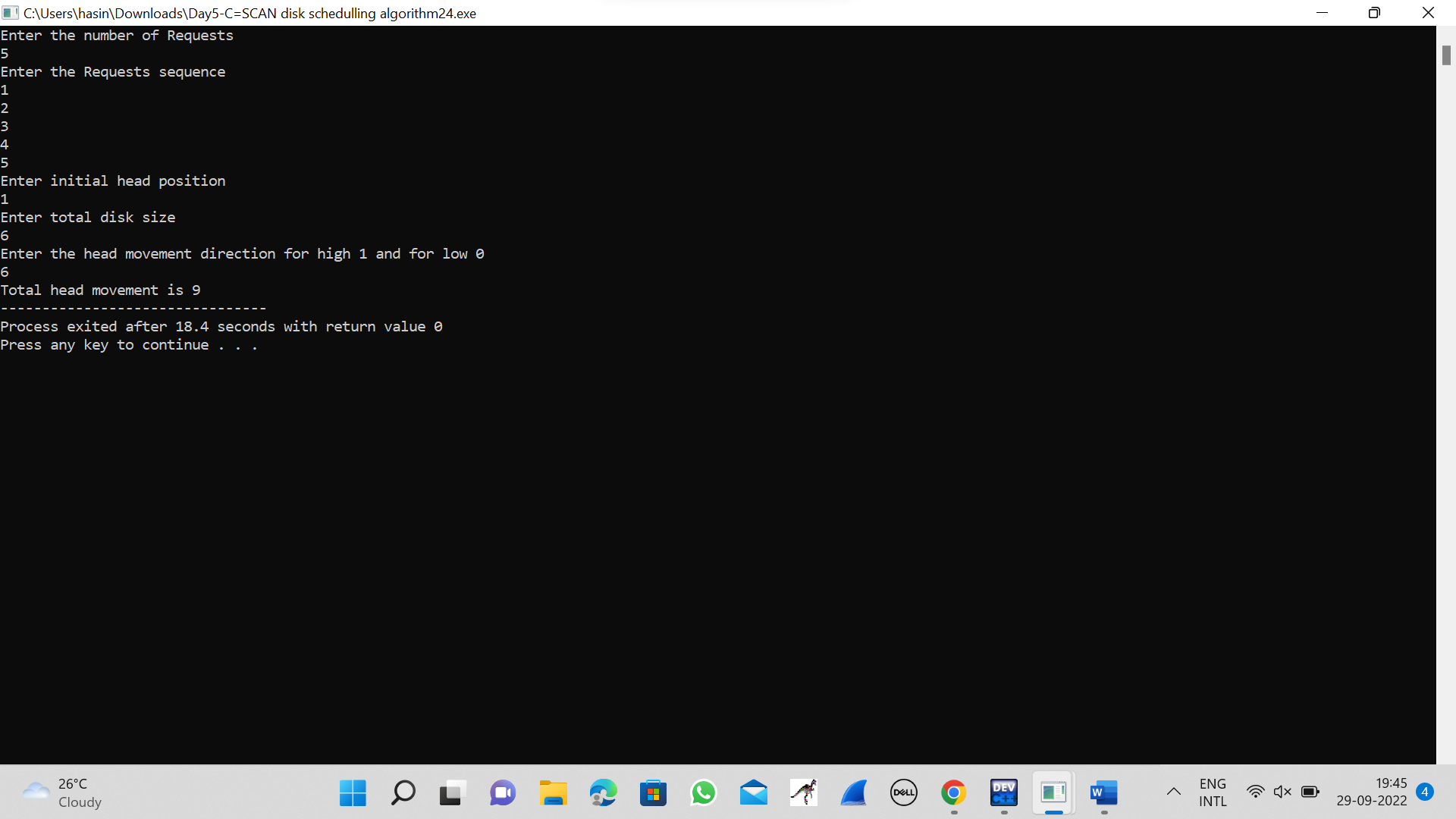
}

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

return 0;

}

**OUTPUT**

****