OS LAB-1BM21CS203

Write a C program to simulate disk scheduling algorithms

a) FCFS

b) SCAN

c) C-SCAN

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


void fcfs(int queue[], int n, int head) {
int totalMovement = 0;

printf("FCFS Scheduling\n");
printf("Sequence of movement: %d ", head);

for (int i = 0; i < n; i++) {
totalMovement += abs(queue[i] - head);
head = queue[i];
printf("-> %d ", head);
}

printf("\nTotal head movement: %d\n\n", totalMovement);
}


void sstf(int queue[], int n, int head) {
int totalMovement = 0;
int visited[n];
```

```
for (int i = 0; i < n; i++) {
visited[i] = 0; //initialise entire visited array to 0(all unvisited initially)
}

printf("SSTF Scheduling\n");
printf("Sequence of movement: %d ", head);

for (int i = 0; i < n; i++) {
int minDistance = 9999;
int index = -1;

for (int j = 0; j < n; j++) {
if (visited[j]==0 && abs(queue[j] - head) < minDistance) {
minDistance = abs(queue[j] - head);
index = j;
}
}

visited[index] = 1;
totalMovement += minDistance;
head = queue[index];
printf("-> %d ", head);
}

printf("\nTotal head movement: %d\n\n", totalMovement);
}

void scan(int queue[], int n, int head, int direction) {
int totalMovement = 0;
```

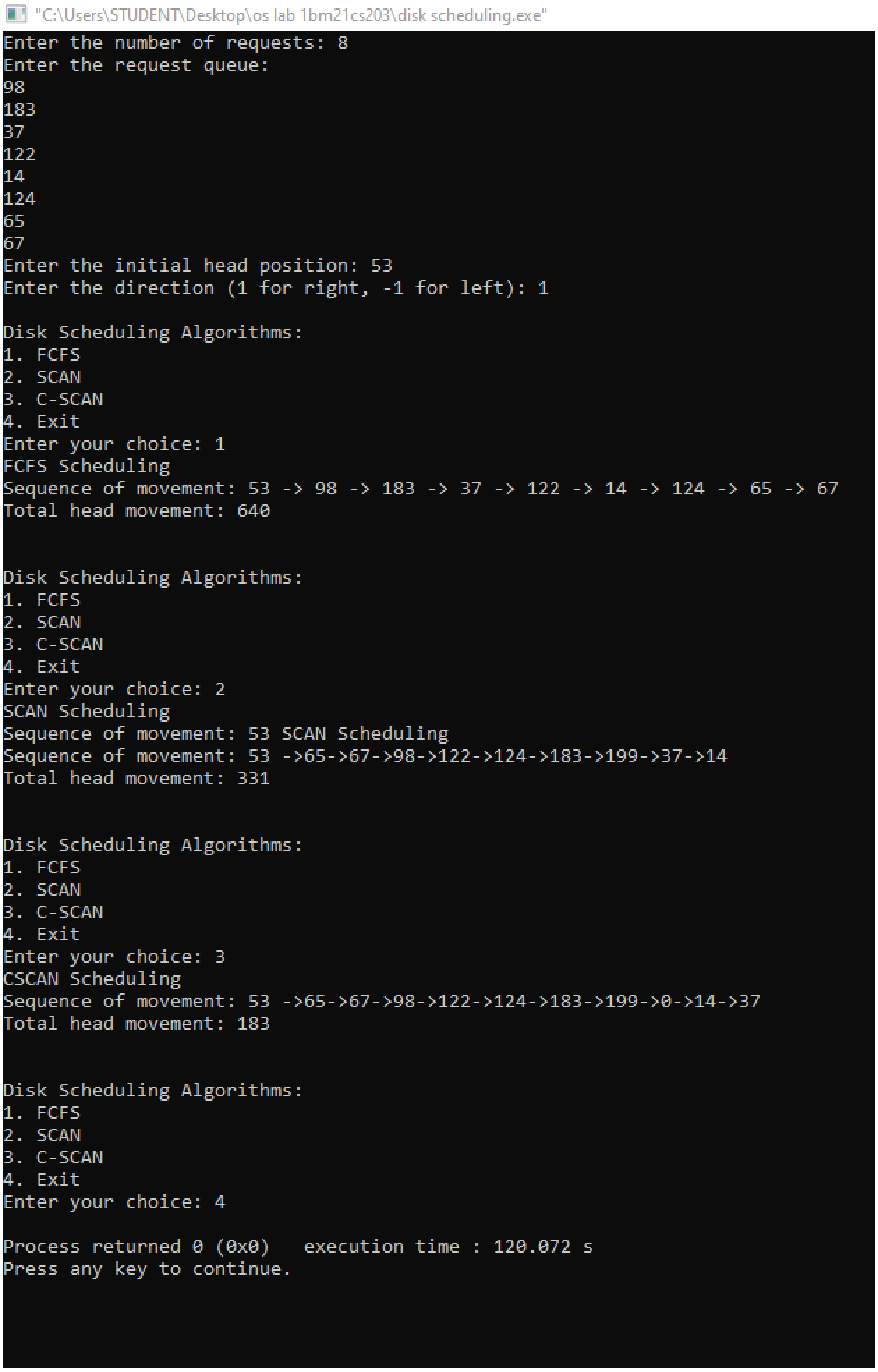
```
printf("SCAN Scheduling\n");
printf("Sequence of movement: %d ", head);

int t1,t2,t3,i;

int pos=0,pos1,pos2=0;//pos of element left of head
for(i=0;i<n;i++)
{
if(queue[i]>head)
{
pos=i-1; break; //pos=1 here
}
}
if(direction==1)
{ printf("SCAN Scheduling\n");
printf("Sequence of movement: %d ", head);
t1=199-head;
pos2=pos;
t3=199-queue[0];

totalMovement=t1+t3;
pos1=pos+1;
while(pos1<=n-1)
printf("->%d",queue[pos1++]);
printf("->199");
while(pos2>=0)

printf("->%d",queue[pos2--]);
}
else
```

```
{
t1=head;
t2=199;
totalMovement=t1+t2;
pos1=pos; pos2=pos+1;
printf("SCAN Scheduling\n");
printf("Sequence of movement: %d ", head);
while(pos1>=0)
printf("->%d",queue[pos1--]);
while(pos2<=n-1)
printf("->%d",queue[pos2++]);
printf("->199");
}

printf("\nTotal head movement: %d\n\n", totalMovement);
}

void cscan(int queue[], int n, int head, int direction) {
int t1,t2,i;
int totalMovement = 0; int pos=0,pos1,pos2=0;//pos of element left of head
for(i=0;i<n;i++)
{
if(queue[i]>head)
{
pos=i-1; break; //pos=1 here
}

}
if(direction==1)
{ printf("CSCAN Scheduling\n");
printf("Sequence of movement: %d ", head);
t1=199-head;
t2=queue[pos];
totalMovement=t1+t2;
pos1=pos+1;
while(pos1<=n-1)
printf("->%d",queue[pos1++]);
printf("->199->0");
while(pos2<=pos)
printf("->%d",queue[pos2++]);
}
else
{
t1=head;
t2=199-queue[pos+1];
totalMovement=t1+t2;
pos1=pos; pos2=n-1;
printf("CSCAN Scheduling\n");
printf("Sequence of movement: %d ", head);
while(pos1>=0)
printf("->%d",queue[pos1--]);
printf("->0->199");
while(pos2>pos)
printf("->%d",queue[pos2--]);
}

printf("\nTotal head movement: %d\n\n", totalMovement);


}
```

```
int main() {
int n, head, direction;

printf("Enter the number of requests: ");
scanf("%d", &n);

int queue[n];

int queue1[n];

printf("Enter the request queue:\n");
for (int i = 0; i < n; i++) {
scanf("%d", &queue[i]);
queue1[i]=queue[i];
}

printf("Enter the initial head position: ");
scanf("%d", &head);
for(int u=0;u<n;u++)
queue1[u]=queue[u];
//sort
for (int k=0;k<n-1;k++)
{
for(int y=0;y<n-k-1;y++)
{
if(queue[y]>queue[y+1])

{
int temp= queue[y];
queue[y]=queue[y+1];
queue[y+1]=temp;
}
}

}

printf("Enter the direction (1 for right, -1 for left): ");
scanf("%d", &direction);

while (1) {
printf("\nDisk Scheduling Algorithms:\n");
printf("1. FCFS\n");
printf("2. SCAN\n");
printf("3. C-SCAN\n");
printf("4. Exit\n");
printf("Enter your choice: ");

int choice;
scanf("%d", &choice);

switch (choice) {
case 1:
fcfs(queue1, n, head);
break;

case 2:

scan(queue, n, head, direction);
break;
case 3:
```

```
cscan(queue, n, head, direction);
break;

case 4:
exit(0);
default:
printf("Invalid choice! Please try again.\n");
}
}

return 0;
}
```

OUTPUT:



```
"C:\Users\STUDENT\Desktop\os lab 1bm21cs203\disk scheduling.exe"
Enter the number of requests: 8
Enter the request queue:
98
183
37
122
14
124
65
67
Enter the initial head position: 53
Enter the direction (1 for right, -1 for left): 1

Disk Scheduling Algorithms:
1. FCFS
2. SCAN
3. C-SCAN
4. Exit
Enter your choice: 1
FCFS Scheduling
Sequence of movement: 53 -> 98 -> 183 -> 37 -> 122 -> 14 -> 124 -> 65 -> 67
Total head movement: 640


Disk Scheduling Algorithms:
1. FCFS
2. SCAN
3. C-SCAN
4. Exit
Enter your choice: 2
SCAN Scheduling
Sequence of movement: 53 SCAN Scheduling
Sequence of movement: 53 ->65->67->98->122->124->183->199->37->14
Total head movement: 331


Disk Scheduling Algorithms:
1. FCFS
2. SCAN
3. C-SCAN
4. Exit
Enter your choice: 3
CSCAN Scheduling
Sequence of movement: 53 ->65->67->98->122->124->183->199->0->14->37
Total head movement: 183


Disk Scheduling Algorithms:
1. FCFS
2. SCAN
3. C-SCAN
4. Exit
Enter your choice: 4

Process returned 0 (0x0)   execution time : 120.072 s
Press any key to continue.
```

Write a C program to simulate disk scheduling algorithms

a) SSTF

b) LOOK

c) c-LOOK

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void sstf(int queue[], int n, int head) {
int totalMovement = 0;
int visited[n];
for (int i = 0; i < n; i++) {
visited[i] = 0; //initialise entire visited array to 0(all unvisited initially)
}
printf("SSTF Scheduling\n");
printf("Sequence of movement: %d ", head);
for (int i = 0; i < n; i++) {
int minDistance = 9999;
int index = -1;
for (int j = 0; j < n; j++) {
if (visited[j]==0 && abs(queue[j] - head) < minDistance) {
minDistance = abs(queue[j] - head);
index = j;
}
}
visited[index] = 1;
totalMovement += minDistance;
head = queue[index];
printf("-> %d ", head);
}
```

```
printf("\nTotal head movement: %d\n\n", totalMovement);
}

void look(int queue[], int n, int head, int direction) {
int totalMovement = 0; int t1=0,t2=0;

int pos=0,pos1,pos2=0;//position of element left of head
for(int i=0;i<n;i++)
{
if(queue[i]>head)
{
pos=i-1; break; //pos=1 here
}
}
printf("LOOK Scheduling\n");
printf("Sequence of movement: %d ", head);
if (direction == 1) {
t1=queue[n-1]-head;
t2=queue[n-1]-queue[0];
pos1=pos+1;
while(pos1<=n-1)
printf("->%d",queue[pos1++]);
pos2=pos;
while(pos2>=0)
printf("->%d",queue[pos2--]);
totalMovement=t1+t2;
}
else {
t1=head-queue[0];
t2=queue[n-1]-queue[0];
```

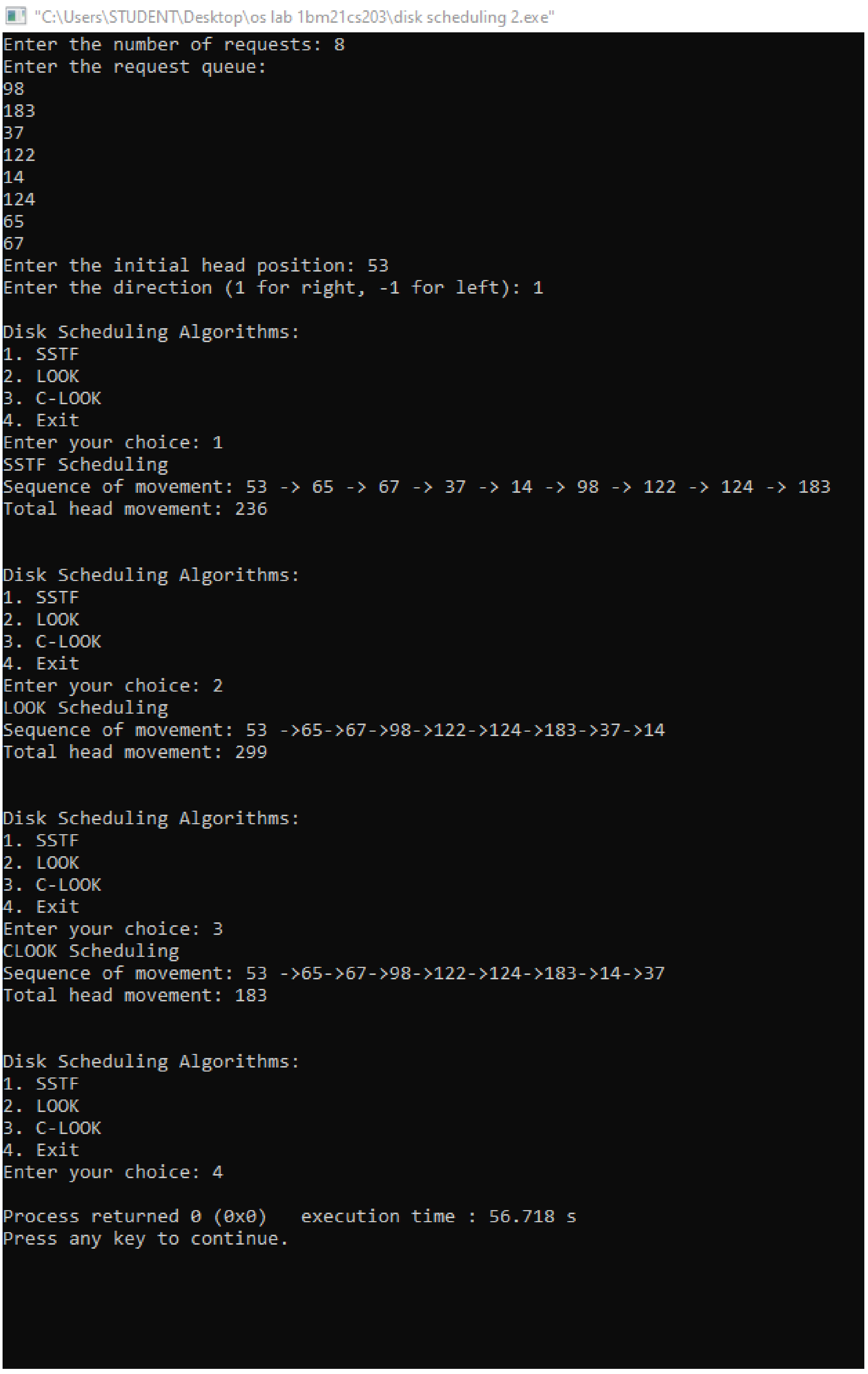
```
totalMovement=t1+t2;
pos1=pos;
while(pos1>=0)
printf("->%d",queue[pos1--]);
pos2=pos+1;
while(pos2<=n-1)
printf("->%d",queue[pos2++]);
}
printf("\nTotal head movement: %d\n\n", totalMovement);
}
void clook(int queue[], int n, int head, int direction) {
int totalMovement = 0; int t1=0,t2=0,t3=0;
int pos=0,pos1,pos2=0;//position of element left of head
for(int i=0;i<n;i++)
{
if(queue[i]>head)
{
pos=i-1; break; //pos=1 here
}

}
printf("CLOOK Scheduling\n");
printf("Sequence of movement: %d ", head);
if (direction == 1) {
t1=queue[n-1]-head;
pos1=pos;
t2=queue[pos1]-queue[0];
t3=(199-queue[n-1])+(queue[0]);
pos1=pos+1;
while(pos1<=n-1)
```

```
printf("->%d",queue[pos1++]);
pos2=0;
while(pos2<=pos)
printf("->%d",queue[pos2++]);
totalMovement=t1+t2+t3;
}
else {
pos1=pos+1;
t1=head-queue[0];
t2=queue[n-1]-queue[pos1];
t3=queue[0]+199-queue[n-1];
totalMovement=t1+t2+t3;
pos1=pos;
while(pos1>=0)
printf("->%d",queue[pos1--]);
pos1=pos+1;
pos2=n-1;
while(pos2>=pos1)
printf("->%d",queue[pos2--]);
}
printf("\nTotal head movement: %d\n\n", totalMovement);
}
int main() {
int n, head, direction;
printf("Enter the number of requests: ");
scanf("%d", &n);
int queue[n];
int queue1[n];
printf("Enter the request queue:\n");
for (int i = 0; i < n; i++) {
```

```
scanf("%d", &queue[i]);
queue1[i]=queue[i];


}
for (int k=0;k<n-1;k++)
{
for(int y=0;y<n-k-1;y++)
{
if(queue[y]>queue[y+1])
{
int temp= queue[y];
queue[y]=queue[y+1];
queue[y+1]=temp;
}
}
}
printf("Enter the initial head position: ");
scanf("%d", &head);
printf("Enter the direction (1 for right, -1 for left): ");
scanf("%d", &direction);
while (1) {
printf("\nDisk Scheduling Algorithms:\n");
printf("1. SSTF\n");
printf("2. LOOK\n");
printf("3. C-LOOK\n");
printf("4. Exit\n");
printf("Enter your choice: ");
int choice;
scanf("%d", &choice);
switch (choice) {
```

```
case 1:
sstf(queue1, n, head);
break;
case 2:
look(queue, n, head, direction);
break;
case 3:
clook(queue, n, head, direction);
break;
case 4:
exit(0);
default:

printf("Invalid choice! Please try again.\n");
}
}
return 0;
}
```

OUTPUT:



"C:\Users\STUDENT\Desktop\os lab 1bm21cs203\disk scheduling 2.exe"

```
Enter the number of requests: 8
Enter the request queue:
98
183
37
122
14
124
65
67
Enter the initial head position: 53
Enter the direction (1 for right, -1 for left): 1

Disk Scheduling Algorithms:
1. SSTF
2. LOOK
3. C-LOOK
4. Exit
Enter your choice: 1
SSTF Scheduling
Sequence of movement: 53 -> 65 -> 67 -> 37 -> 14 -> 98 -> 122 -> 124 -> 183
Total head movement: 236


Disk Scheduling Algorithms:
1. SSTF
2. LOOK
3. C-LOOK
4. Exit
Enter your choice: 2
LOOK Scheduling
Sequence of movement: 53 ->65->67->98->122->124->183->37->14
Total head movement: 299


Disk Scheduling Algorithms:
1. SSTF
2. LOOK
3. C-LOOK
4. Exit
Enter your choice: 3
CLOOK Scheduling
Sequence of movement: 53 ->65->67->98->122->124->183->14->37
Total head movement: 183


Disk Scheduling Algorithms:
1. SSTF
2. LOOK
3. C-LOOK
4. Exit
Enter your choice: 4

Process returned 0 (0x0)   execution time : 56.718 s
Press any key to continue.
```

Write a C program to simulate page replacement algorithms

a) FIFO

b) LRU

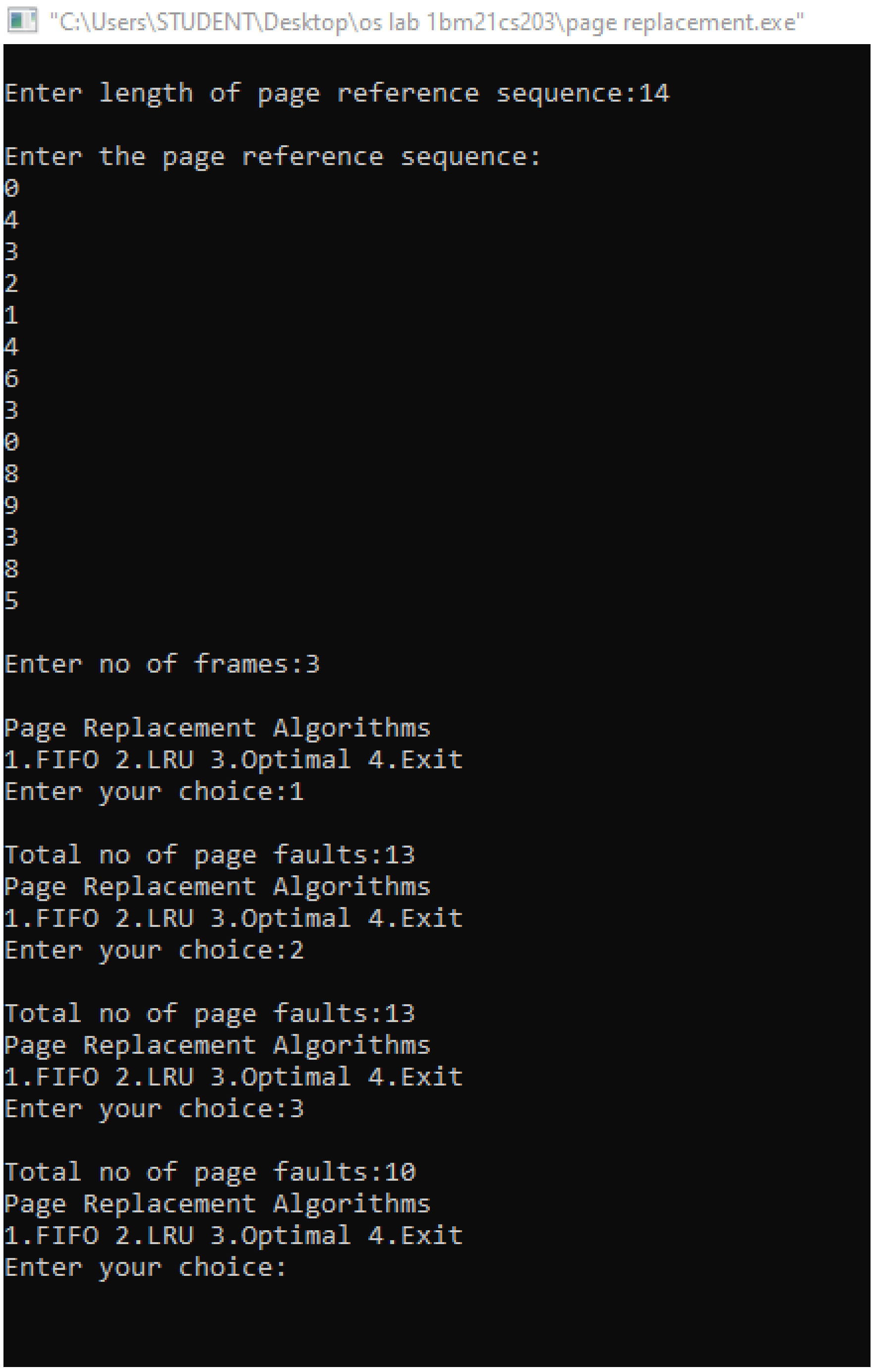
c) Optimal

```
#include<stdio.h>
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;


void initialize()
{
    pgfaultcnt=0;
    for(i=0; i<nf; i++)
        p[i]=9999;
}

int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }
    }

    return hit;
}

int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}

void dispPgFaultCnt()
{
    printf("\nTotal no of page faults:%d",pgfaultcnt);
}

void fifo()
{
    initialize();
    for(i=0; i<n; i++)
    {
        if(isHit(in[i])==0)
        {

            for(k=0; k<nf-1; k++)
                p[k]=p[k+1];

            p[k]=in[i];
            pgfaultcnt++;

        }

    }
    dispPgFaultCnt();
}


void optimal()
{
    initialize();
    int near[50];
    for(i=0; i<n; i++)
    {


        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
```

```
    int pg=p[j];
    int found=0;
    for(k=i; k<n; k++)
    {
        if(pg==in[k])
        {
            near[j]=k;
            found=1;
            break;
        }
        else
            found=0;
    }
    if(!found)
        near[j]=9999;
}
int max=-9999;
int repindex;
for(j=0; j<nf; j++)
{
    if(near[j]>max)
    {
        max=near[j];
        repindex=j;
    }
}
p[repindex]=in[i];
pgfaultcnt++;
```

```
        }

    }
    dispPgFaultCnt();
}

void lru()
{
    initialize();

    int least[50];
    for(i=0; i<n; i++)
    {



        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i-1; k>=0; k--)
                {
                    if(pg==in[k])
                    {
                        least[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    least[j]=-9999;
            }
            int min=9999;
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(least[j]<min)
                {
                    min=least[j];
                    repindex=j;
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;


        }


    }
    dispPgFaultCnt();
}



int main()
{
    int choice;
    printf("\nEnter length of page reference sequence:");
    scanf("%d",&n);
    printf("\nEnter the page reference sequence:\n");
    for(i=0; i<n; i++)
        scanf("%d",&in[i]);
    printf("\nEnter no of frames:");
    scanf("%d",&nf);
    while(1)
    {
        printf("\nPage Replacement Algorithms\n1.FIFO 2.LRU 3.Optimal 4.Exit\nEnter your
choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            fifo();
            break;
        case 2:
            lru();
            break;
        case 3:
            optimal();
            break;
        case 4:
            exit(0);
        default:printf("invalid choice");

        }
    }
}
```

OUTPUT:



"C:\Users\STUDENT\Desktop\os lab 1bm21cs203\page replacement.exe"

```
Enter length of page reference sequence:14

Enter the page reference sequence:
0
4
3
2
1
4
6
3
0
8
9
3
8
5

Enter no of frames:3

Page Replacement Algorithms
1.FIFO 2.LRU 3.Optimal 4.Exit
Enter your choice:1

Total no of page faults:13
Page Replacement Algorithms
1.FIFO 2.LRU 3.Optimal 4.Exit
Enter your choice:2

Total no of page faults:13
Page Replacement Algorithms
1.FIFO 2.LRU 3.Optimal 4.Exit
Enter your choice:3

Total no of page faults:10
Page Replacement Algorithms
1.FIFO 2.LRU 3.Optimal 4.Exit
Enter your choice:
```