FALL – SEMESTER
Course Code: MCSE504P
Course-Title: – Operating System
DIGITAL ASSIGNMENT - 6
(LAB)

Name: Nidhi Singh
Reg. No:22MAI0015
Slot- L51+L52

Faculty : SALEEM DURAI M.A - SCOPE

Page Replacement Algorithm and disk scheduling algorithm

(i). Page Replacement

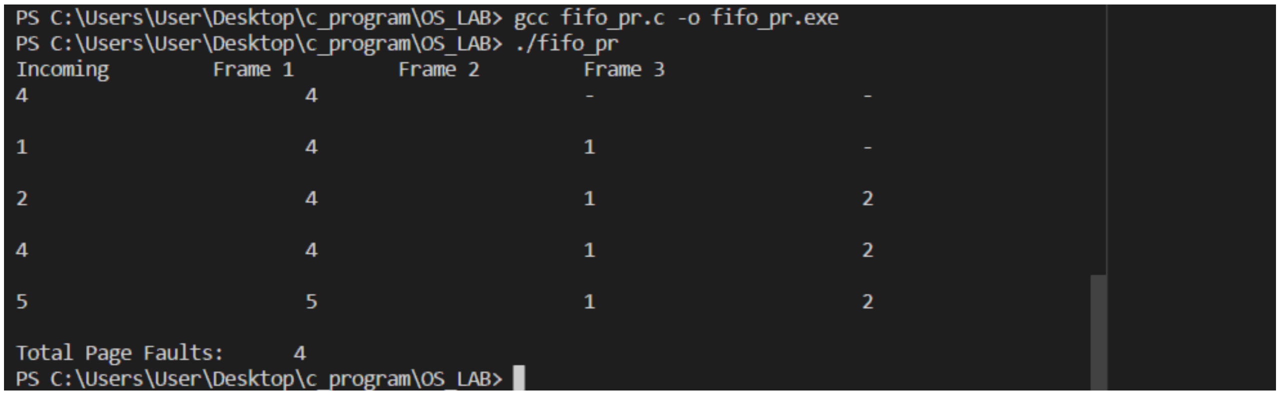
1. First In First Out (FIFO)
2. Optimal page replacement
3. Least Recently Used (LRU) Page Replacement Algorithm

1. First In First Out (FIFO)

Code :-

```
#include <stdio.h>
int main()
{
    int incomingStream[] = {4, 1, 2, 4, 5};
    int pageFaults = 0;
    int frames = 3;
    int m, n, s, pages;

    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

    printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");
    int temp[frames];
    for(m = 0; m < frames; m++)
    {
        temp[m] = -1;
    }

    for(m = 0; m < pages; m++)
    {
        s = 0;

        for(n = 0; n < frames; n++)
        {
            if(incomingStream[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
        pageFaults++;

        if((pageFaults <= frames) && (s == 0))
        {
            temp[m] = incomingStream[m];
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }

        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
    }

    printf("\nTotal Page Faults:\t%d\n", pageFaults);
    return 0;
}
```

Output :-



```
PS C:\Users\User\Desktop\c_program\OS_LAB> gcc fifo_pr.c -o fifo_pr.exe
PS C:\Users\User\Desktop\c_program\OS_LAB> ./fifo_pr
Incoming        Frame 1        Frame 2        Frame 3
4                   4                  -                  -

1                   4                  1                  -

2                   4                  1                  2

4                   4                  1                  2

5                   5                  1                  2

Total Page Faults:     4
PS C:\Users\User\Desktop\c_program\OS_LAB>
```

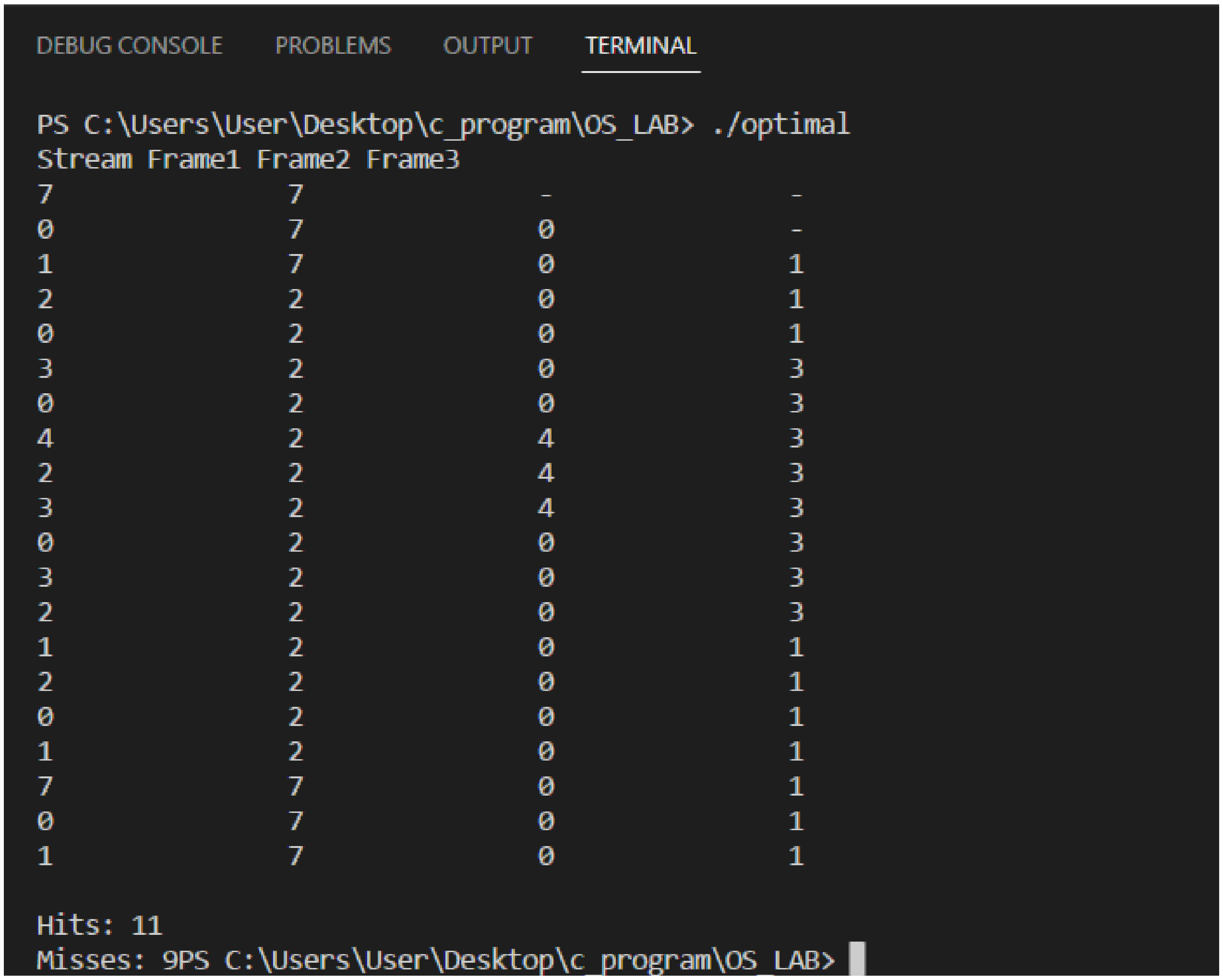
2. Optimal page replacement

Code :-

```
#include <stdio.h>
 int search(int key, int frame_items[], int frame_occupied)
{
    for (int i = 0; i < frame_occupied; i++)
        if (frame_items[i] == key)
            return 1;
    return 0;
}

void printOuterStructure(int max_frames){
    printf("Stream ");

    for(int i = 0; i < max_frames; i++)
        printf("Frame%d ", i+1);
}
void printCurrFrames(int item, int frame_items[], int frame_occupied, int max_frames){

    // print current reference stream item
    printf("\n%d \t\t", item);

    // print frame occupants one by one
    for(int i = 0; i < max_frames; i++){
        if(i < frame_occupied)
            printf("%d \t\t", frame_items[i]);
        else
            printf("- \t\t");
    }
}

int predict(int ref_str[], int frame_items[], int refStrLen, int index, int frame_occupied)
```

```
{
     int result = -1, farthest = index;
    for (int i = 0; i < frame_occupied; i++) {
        int j;
        for (j = index; j < refStrLen; j++)
        {
            if (frame_items[i] == ref_str[j])
            {
                if (j > farthest) {
                    farthest = j;
                    result = i;
                }
                break;
            }
        }
     if (j == refStrLen)
            return i;
    }
   return (result == -1) ? 0 : result;
}

void optimalPage(int ref_str[], int refStrLen, int frame_items[], int max_frames)
{
     int frame_occupied = 0;
    printOuterStructure(max_frames);
     int hits = 0;
    for (int i = 0; i < refStrLen; i++) {
    if (search(ref_str[i], frame_items, frame_occupied)) {
            hits++;
            printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
            continue;
        }
     if (frame_occupied < max_frames){
            frame_items[frame_occupied] = ref_str[i];
            frame_occupied++;
            printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
        }
         else {
            int pos = predict(ref_str, frame_items, refStrLen, i + 1, frame_occupied);
            frame_items[pos] = ref_str[i];
            printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
        }

    }
    printf("\n\nHits: %d\n", hits);
    printf("Misses: %d", refStrLen - hits);
}

int main()
{
    int ref_str[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};
    int refStrLen = sizeof(ref_str) / sizeof(ref_str[0]);
    int max_frames = 3;
    int frame_items[max_frames];

    optimalPage(ref_str, refStrLen, frame_items, max_frames);
    return 0;
}
```

Output :-



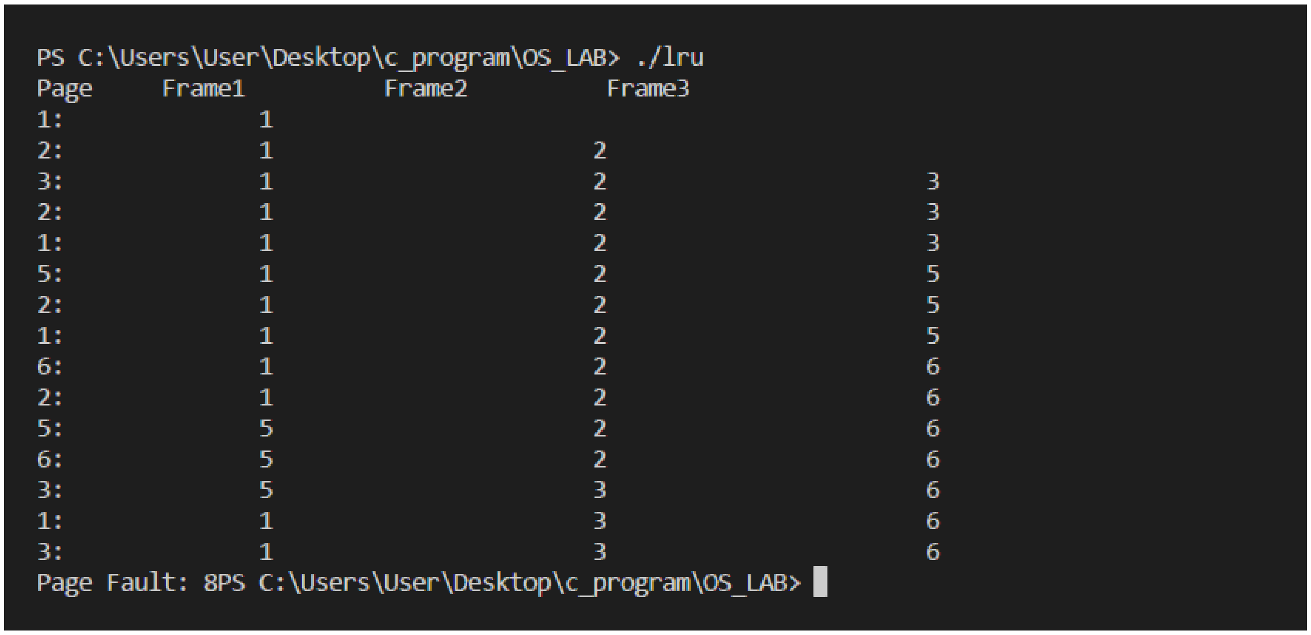
```
PS C:\Users\User\Desktop\c_program\OS_LAB> ./optimal
Stream Frame1 Frame2 Frame3
7             7            -            -
0             7            0            -
1             7            0            1
2             2            0            1
0             2            0            1
3             2            0            3
0             2            0            3
4             2            4            3
2             2            4            3
3             2            4            3
0             2            0            3
3             2            0            3
2             2            0            3
1             2            0            1
2             2            0            1
0             2            0            1
1             2            0            1
7             7            0            1
0             7            0            1
1             7            0            1

Hits: 11
Misses: 9PS C:\Users\User\Desktop\c_program\OS_LAB>
```

3. Least Recently Used (LRU) Page Replacement Algorithm

Code :-

```
#include<stdio.h>
#include<limits.h>
int checkHit(int incomingPage, int queue[], int occupied){
    for(int i = 0; i < occupied; i++){
        if(incomingPage == queue[i])
            return 1;
    }
    return 0;
}
void printFrame(int queue[], int occupied)
{
    for(int i = 0; i < occupied; i++)
        printf("%d\t\t\t",queue[i]);
}
int main()
{
   int incomingStream[] = {1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3};
    int n = sizeof(incomingStream)/sizeof(incomingStream[0]);
    int frames = 3;
    int queue[n];
    int distance[n];
    int occupied = 0;
    int pagefault = 0;
    printf("Page\t Frame1 \t Frame2 \t Frame3\n");
    for(int i = 0;i < n; i++)
    {
        printf("%d:  \t\t",incomingStream[i]);
         if(checkHit(incomingStream[i], queue, occupied)){
            printFrame(queue, occupied);
        }

else if(occupied < frames){
            queue[occupied] = incomingStream[i];
            pagefault++;
            occupied++;
             printFrame(queue, occupied);
        }
        else{
             int max = INT_MIN;
            int index;
             for (int j = 0; j < frames; j++)
            {
                distance[j] = 0;
                 for(int k = i - 1; k >= 0; k--)
                {
                    ++distance[j];
          if(queue[j] == incomingStream[k])
                        break;
                }
                 if(distance[j] > max){
                    max = distance[j];
                    index = j;
                }
            }
            queue[index] = incomingStream[i];
            printFrame(queue, occupied);
            pagefault++;
        }
         printf("\n");
    }

    printf("Page Fault: %d",pagefault);

    return 0;
}
```

Output :-



```
PS C:\Users\User\Desktop\c_program\OS_LAB> ./lru
Page     Frame1          Frame2          Frame3
1:              1
2:              1                       2
3:              1                       2                       3
2:              1                       2                       3
1:              1                       2                       3
5:              1                       2                       5
2:              1                       2                       5
1:              1                       2                       5
6:              1                       2                       6
2:              1                       2                       6
5:              5                       2                       6
6:              5                       2                       6
3:              5                       3                       6
1:              1                       3                       6
3:              1                       3                       6
Page Fault: 8PS C:\Users\User\Desktop\c_program\OS_LAB> █
```

Types of disk scheduling algorithms **are:**

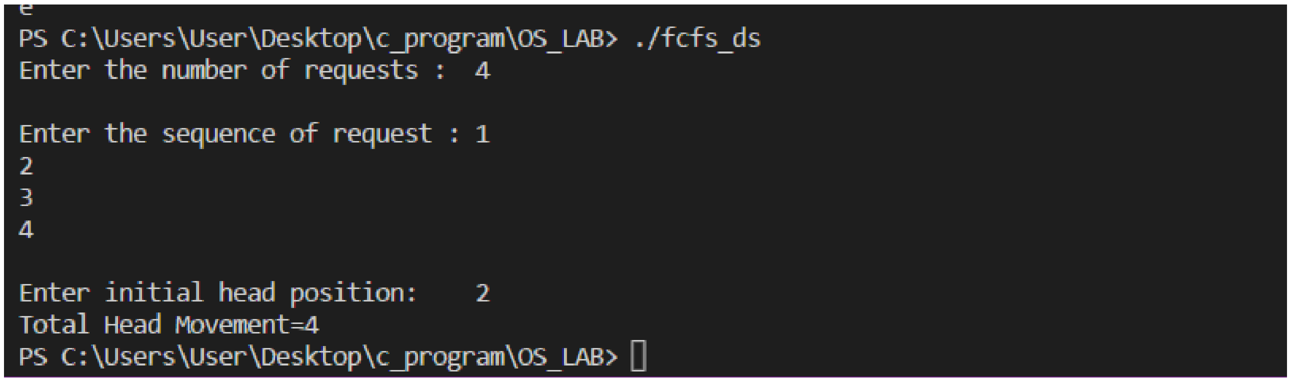
1. FCFS (First come first serve) algorithm
2. SSTF (Shortest seek time first) algorithm
3. Scan Disk Algorithm
4. C-Scan (Circular scan) algorithm
5. Look algorithm
6. C-Look (Circular look) algorithm

FCFS (First come first serve) algorithm

Code :-

```
 #include<stdio.h>
#include<stdlib.h>
int main()
{
  int ReadyQueue[100],i,n,TotalHeadMov=0,initial;
  printf("Enter the number of requests :\t");
  scanf("%d",&n);
  printf("\nEnter the sequence of request :\t");
  for(i=0;i<n;i++){
    scanf("%d",&ReadyQueue[i]);
  }
  printf("\nEnter initial head position:\t");
  scanf("%d",&initial);
  for(i=0;i<n;i++)
  {
    TotalHeadMov=TotalHeadMov+abs(ReadyQueue[i]-initial);
    initial=ReadyQueue[i];
  }
  printf("Total Head Movement=%d\n",TotalHeadMov);
}
```

Output :-



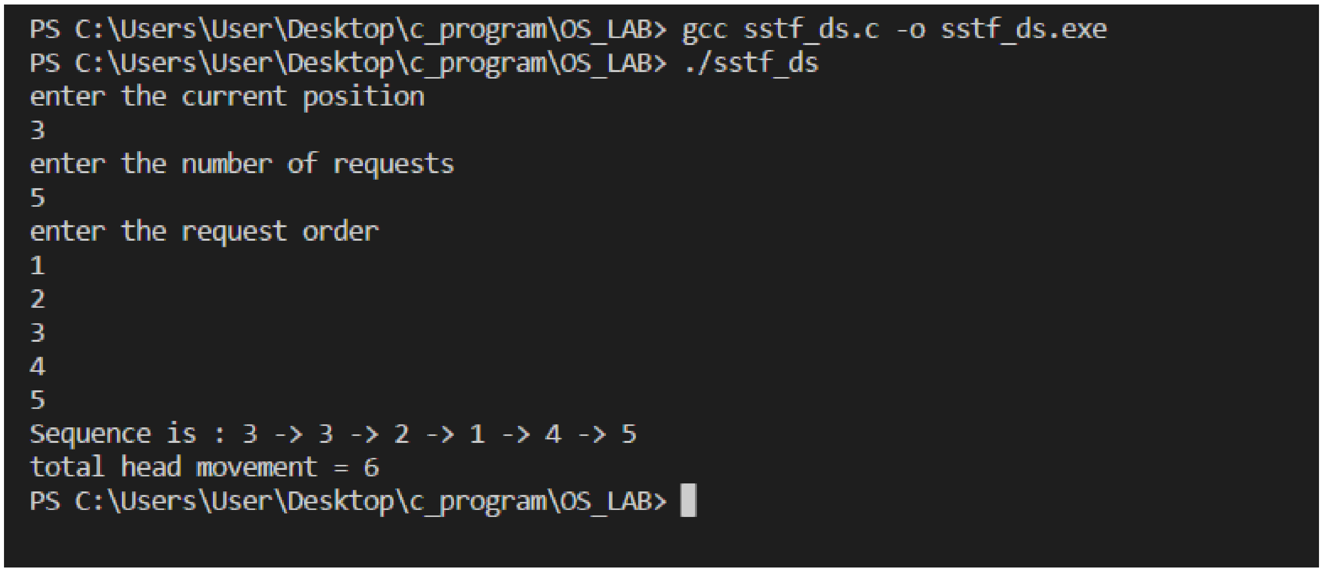
```
PS C:\Users\User\Desktop\c_program\OS_LAB> ./fcfs_ds
Enter the number of requests :  4

Enter the sequence of request : 1
2
3
4

Enter initial head position:    2
Total Head Movement=4
PS C:\Users\User\Desktop\c_program\OS_LAB> 
```

SSTF (Shortest seek time first) algorithm

Code :-

```
 #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]);
    }
     min=index[0];
    mini=0;
    for(i=1;i<n;i++)
    {
        if(min>index[i])
        {
            min=index[i];
            mini=i;
        }
    }
    a[j]=req[mini];
    j++;
    cp=req[mini];
   req[mini]=999;
    }
   printf("Sequence is : ");
    printf("%d",cp1);
    mov=mov+abs(cp1-a[0]);
    printf(" -> %d",a[0]);
    for(i=1;i<n;i++)
    {
        mov=mov+abs(a[i]-a[i-1]);
        printf(" -> %d",a[i]);
    }
    printf("\n");
    printf("total head movement = %d\n",mov);
}
```

Output :-



```
PS C:\Users\User\Desktop\c_program\OS_LAB> gcc sstf_ds.c -o sstf_ds.exe
PS C:\Users\User\Desktop\c_program\OS_LAB> ./sstf_ds
enter the current position
3
enter the number of requests
5
enter the request order
1
2
3
4
5
Sequence is : 3 -> 3 -> 2 -> 1 -> 4 -> 5
total head movement = 6
PS C:\Users\User\Desktop\c_program\OS_LAB> 
```

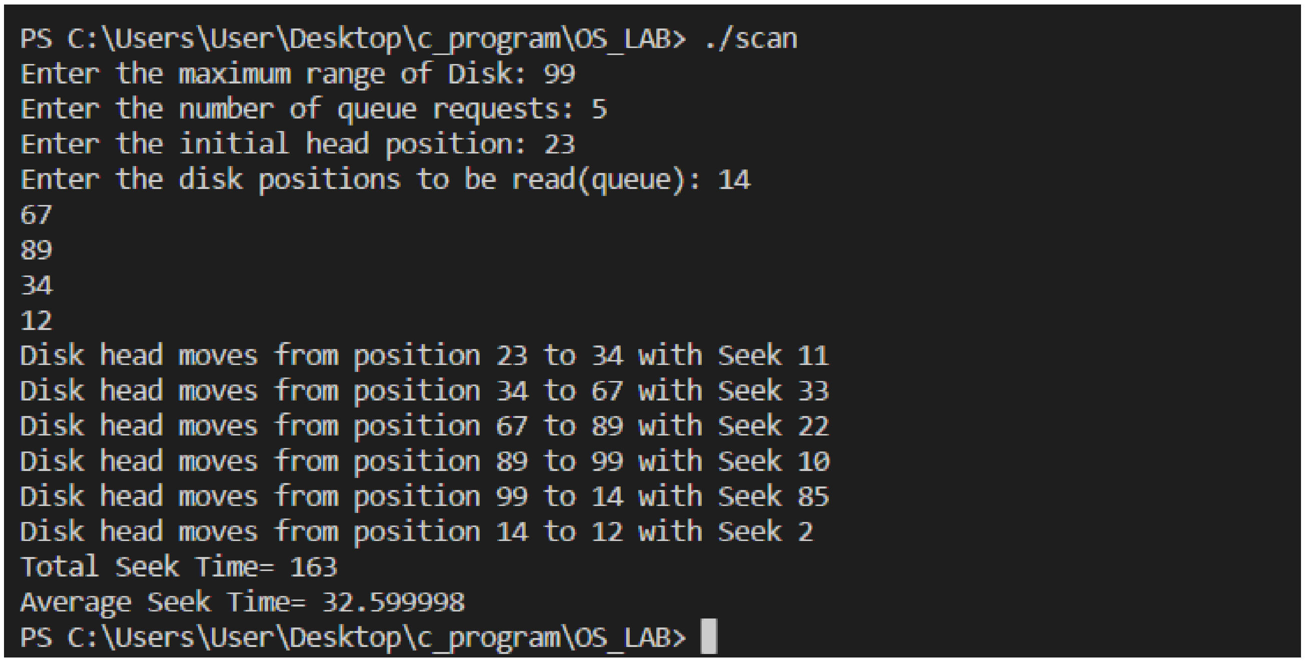
Scan Disk Algorithm

Code :-

```
#include<stdio.h>
int absoluteValue(int); // Declaring function absoluteValue

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;
   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++)   // Note that i varies from 1 to n instead of 0 to n-1
    {
        scanf("%d",&temp);  //Reading position value to a temporary variable
         if(temp>headposition)
        {
            queue1[temp1]=temp; //temp1 is the index variable of queue1[]
            temp1++; //incrementing temp1
        }
        else    //else if temp < current headposition,then push to array queue2[]
        {
            queue2[temp2]=temp; //temp2 is the index variable of queue2[]
            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]=maxrange;

     for(i=temp1+2,j=0;j<temp2;i++,j++)
    {
        queue[i]=queue2[j];
    }
     queue[i]=0;

     queue[0]=headposition;
     for(j=0; j<=n; j++) //Loop starts from headposition. (ie. 0th index of queue)
    {
         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)
    {
        return x;
    }
    else
    {
        return x*-1;
    }
```

```
}
```

Output :-



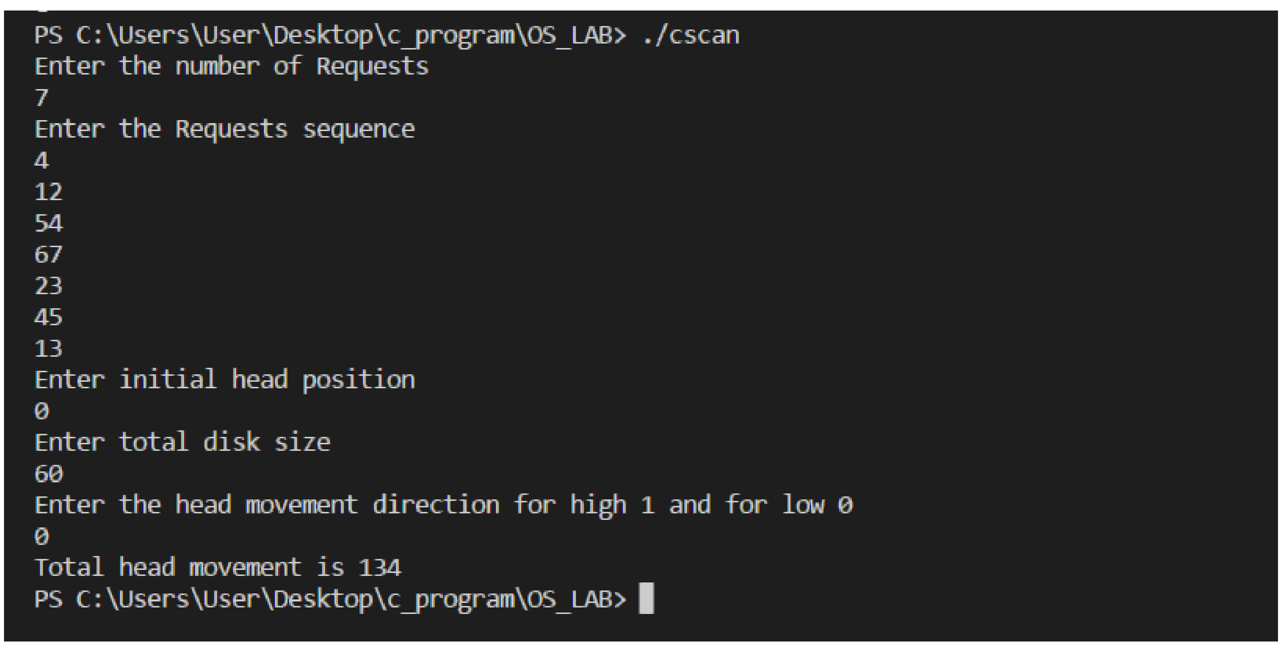
```
PS C:\Users\User\Desktop\c_program\OS_LAB> ./scan
Enter the maximum range of Disk: 99
Enter the number of queue requests: 5
Enter the initial head position: 23
Enter the disk positions to be read(queue): 14
67
89
34
12
Disk head moves from position 23 to 34 with Seek 11
Disk head moves from position 34 to 67 with Seek 33
Disk head moves from position 67 to 89 with Seek 22
Disk head moves from position 89 to 99 with Seek 10
Disk head moves from position 99 to 14 with Seek 85
Disk head moves from position 14 to 12 with Seek 2
Total Seek Time= 163
Average Seek Time= 32.599998
PS C:\Users\User\Desktop\c_program\OS_LAB>
```

C-Scan (Circular scan) algorithm

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 (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 :-



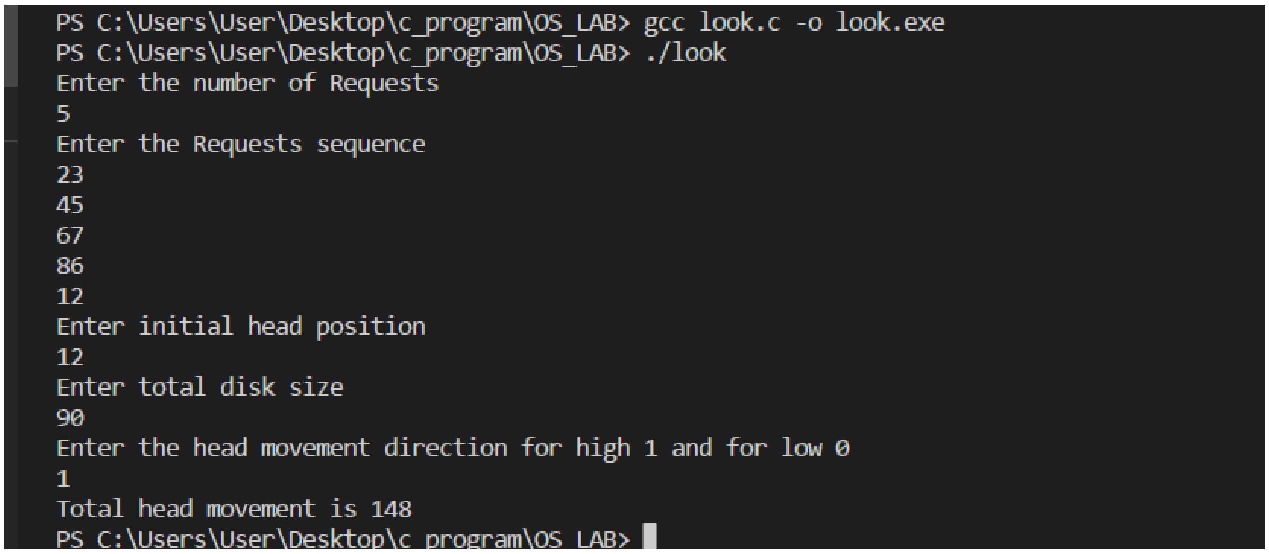
```
PS C:\Users\User\Desktop\c_program\OS_LAB> ./cscan
Enter the number of Requests
7
Enter the Requests sequence
4
12
54
67
23
45
13
Enter initial head position
0
Enter total disk size
60
Enter the head movement direction for high 1 and for low 0
0
Total head movement is 134
PS C:\Users\User\Desktop\c_program\OS_LAB>
```

Look algorithm

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 (move == 1){
        for (i = index; i < n; i++){
            TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
            initial = RQ[i];
        }
        for (i = index - 1; i >= 0; 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];
        }
        for (i = index; i < n; i++){
            TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
            initial = RQ[i];
        }
    }
    printf("Total head movement is %d \n", TotalHeadMoment);
    return 0;
}
```

Output :-



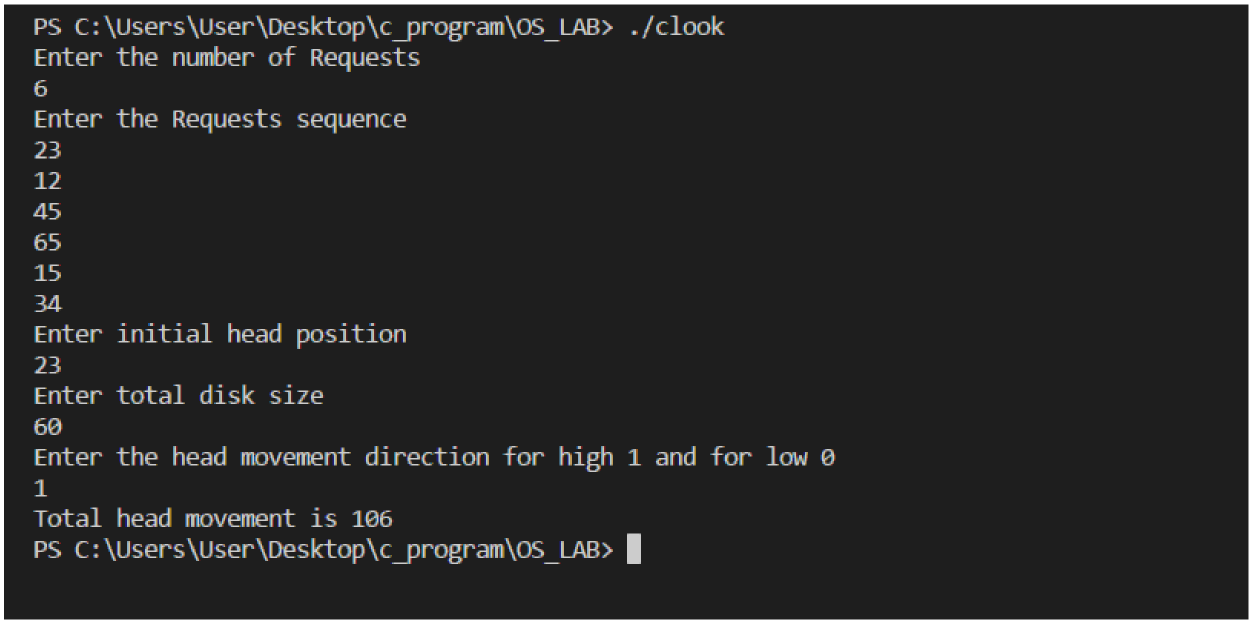
```
PS C:\Users\User\Desktop\c_program\OS_LAB> gcc look.c -o look.exe
PS C:\Users\User\Desktop\c_program\OS_LAB> ./look
Enter the number of Requests
5
Enter the Requests sequence
23
45
67
86
12
Enter initial head position
12
Enter total disk size
90
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 148
PS C:\Users\User\Desktop\c_program\OS_LAB>
```

C-Look (Circular look) algorithm

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(move==1)
    {
        for(i=index;i<n;i++)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for( i=0;i<index;i++)
        {
             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=n-1;i>=index;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];

        }
    }

    printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

Output :-



```
PS C:\Users\User\Desktop\c_program\OS_LAB> ./clook
Enter the number of Requests
6
Enter the Requests sequence
23
12
45
65
15
34
Enter initial head position
23
Enter total disk size
60
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 106
PS C:\Users\User\Desktop\c_program\OS_LAB>
```