

## **Assessment- 4**

### **Winter Sem 2020-21**

**Name : Jogalekar Ishan Sagar**

**Reg no : 19BCE2250**

**Course : CSE2005 - Operating Systems Lab**

**Slot : L35+L36**

### **Memory Management**

(a) Consider a memory hole of size 1kb initially. When a sequence of memory request arrives as following, illustrate the memory allocation by various approaches and calculate the total amount memory wasted by external fragmentation and internal fragmentation in each approach.

I) First fit    II) Best fit    III) Worst fit

Ans :

Code :-

```
#include <iostream>
using namespace std;
int main()
{
    int c,i,j,k,n,l,m[10],p[10],po[20],flag,z,y,temp,temp1;
    cout<<"\n 19BCE2250 - Ishan Jogalekar "<<endl;
    cout<<"Enter memory total partitions:\t";
    cin>>n;
    cout<<"\nEnter memory size for\n";
    for(i=1;i<=n;i++)
```

```

{
    cout<<"\npartition "<<i<<" :\t";
    cin>>m[i];
    po[i]=i;
}

cout<<"\nEnter total number of process:\t";
cin>>j;
cout<<"\nEnter memory size for\n";
for(i=1;i<=j;i++)
{
    cout<<"\nprocess "<<i<<" :\t";
    cin>>p[i];
}

cout<<"\n**Menu**\n1.first fit\n2.best fit\n3.worst fit\nEnter
choice:\t";
cin>>c;
switch(c)
{
case 1:
    for(i=1;i<=j;i++)
    {
        flag=1;
        for(k=1;k<=n;k++)
        {
            if(p[i]<=m[k])

```

```

        {
            cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
allocated at memory partition:\t"<<po[k];

            m[k]=m[k]-p[i];

            break;

        }

        else

        {

            flag++;

        }

    }

    if(flag>n)

    {

        cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
can't be allocated";

    }

}

break;

case 2:

for(y=1;y<=n;y++)

{

    for(z=y;z<=n;z++)

    {

        if(m[y]>m[z])

        {

```

```

temp=m[y];
m[y]=m[z];
m[z]=temp;
temp1=po[y];
po[y]=po[z];
po[z]=temp1;
}
}
}
for(i=1;i<=j;i++)
{
    flag=1;
    for(k=1;k<=n;k++)
    {
        if(p[i]<=m[k])
        {
            cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
allocated at memory partition:\t"<<po[k];
            m[k]=m[k]-p[i];
            break;
        }
        else
        {
            flag++;
        }
    }
}

```

```

    }
    if(flag>n)
    {
        cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
can't be allocated";
    }
}

break;
case 3:
for(y=1;y<=n;y++)
{
    for(z=y;z<=n;z++)
    {
        if(m[y]<m[z])
        {
            temp=m[y];
            m[y]=m[z];
            m[z]=temp;
            temp1=po[y];
            po[y]=po[z];
            po[z]=temp1;
        }
    }
}

for(i=1;i<=j;i++)

```

```

{
    flag=1;
    for(k=1;k<=n;k++)
    {
        if(p[i]<=m[k])
        {
            cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
allocated at memory partition:\t"<<po[k];
            m[k]=m[k]-p[i];
            break;
        }
        else
        {
            flag++;
        }
    }
    if(flag>n)
    {
        cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
can't be allocated";
    }
}

break;

}

return 0;}

```

Output :-

I) First Fit :

```
C:\Windows\System32\cmd.exe

19BCE2250 - Ishan Jogalekar
Enter memory total partitions: 5

Enter memory size for

partition 1 : 50
partition 2 : 60
partition 3 : 70
partition 4 : 80
partition 5 : 90

Enter total number of process: 5

Enter memory size for

process 1 : 10
process 2 : 20
process 3 : 30
process 4 : 40
process 5 : 50

**Menu**
1.first fit
2.best fit
3.worst fit
Enter choice: 1

Process 1 whose memory size is 10KB allocated at memory partition: 1
Process 2 whose memory size is 20KB allocated at memory partition: 1
Process 3 whose memory size is 30KB allocated at memory partition: 2
Process 4 whose memory size is 40KB allocated at memory partition: 3
Process 5 whose memory size is 50KB allocated at memory partition: 4
C:\Users\Dell\Desktop\OS LAB>
```

II) Best Fit :

```

C:\Windows\System32\cmd.exe
C:\Users\Dell\Desktop\OS LAB>g++ 1.cpp

C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter memory total partitions: 5

Enter memory size for

partition 1 : 50

partition 2 : 60

partition 3 : 70

partition 4 : 80

partition 5 : 90

Enter total number of process: 5

Enter memory size for

process 1 : 10

process 2 : 20

process 3 : 30

process 4 : 40

process 5 : 50

**Menu**
1.first fit
2.best fit
3.worst fit
Enter choice: 2

Process 1 whose memory size is 10KB allocated at memory partition: 1
Process 2 whose memory size is 20KB allocated at memory partition: 1
Process 3 whose memory size is 30KB allocated at memory partition: 2
Process 4 whose memory size is 40KB allocated at memory partition: 3
Process 5 whose memory size is 50KB allocated at memory partition: 4
C:\Users\Dell\Desktop\OS LAB>

```

### III) Worst Fit :

```

C:\Windows\System32\cmd.exe
C:\Users\Dell\Desktop\OS LAB>g++ 1.cpp

C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter memory total partitions: 5

Enter memory size for

partition 1 : 50

partition 2 : 60

partition 3 : 70

partition 4 : 80

partition 5 : 90

Enter total number of process: 5

Enter memory size for

process 1 : 10

process 2 : 20

process 3 : 30

process 4 : 40

process 5 : 50

**Menu**
1.first fit
2.best fit
3.worst fit
Enter choice: 3

Process 1 whose memory size is 10KB allocated at memory partition: 5
Process 2 whose memory size is 20KB allocated at memory partition: 5
Process 3 whose memory size is 30KB allocated at memory partition: 5
Process 4 whose memory size is 40KB allocated at memory partition: 4
Process 5 whose memory size is 50KB allocated at memory partition: 3
C:\Users\Dell\Desktop\OS LAB>

```



(b) Write a program to implement the page replacement algorithms.

I. FIFO :

Ans :

Code –

```
#include<stdio.h>

int main()
{
    printf("19BCE2250 - Ishan Jogalekar");
    int reference_string[10], page_faults = 0, m, n, s, pages, frames;
    printf("\nEnter Total Number of Pages:\t");
    scanf("%d", &pages);
    printf("\nEnter values of Reference String:\n");
    for(m = 0; m < pages; m++)
    {
        printf("Value No. [%d]:\t", m + 1);
        scanf("%d", &reference_string[m]);
    }
    printf("\nEnter Total Number of Frames:\t");
    {
        scanf("%d", &frames);
    }
    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(reference_string[m] == temp[n])
        {
            s++;
            page_faults--;
        }
    }
    page_faults++;
    if((page_faults <= frames) && (s == 0))
    {
        temp[m] = reference_string[m];
    }
    else if(s == 0)
    {
        temp[(page_faults - 1) % frames] = reference_string[m];
    }
    printf("\n");
    for(n = 0; n < frames; n++)
    {
```

```

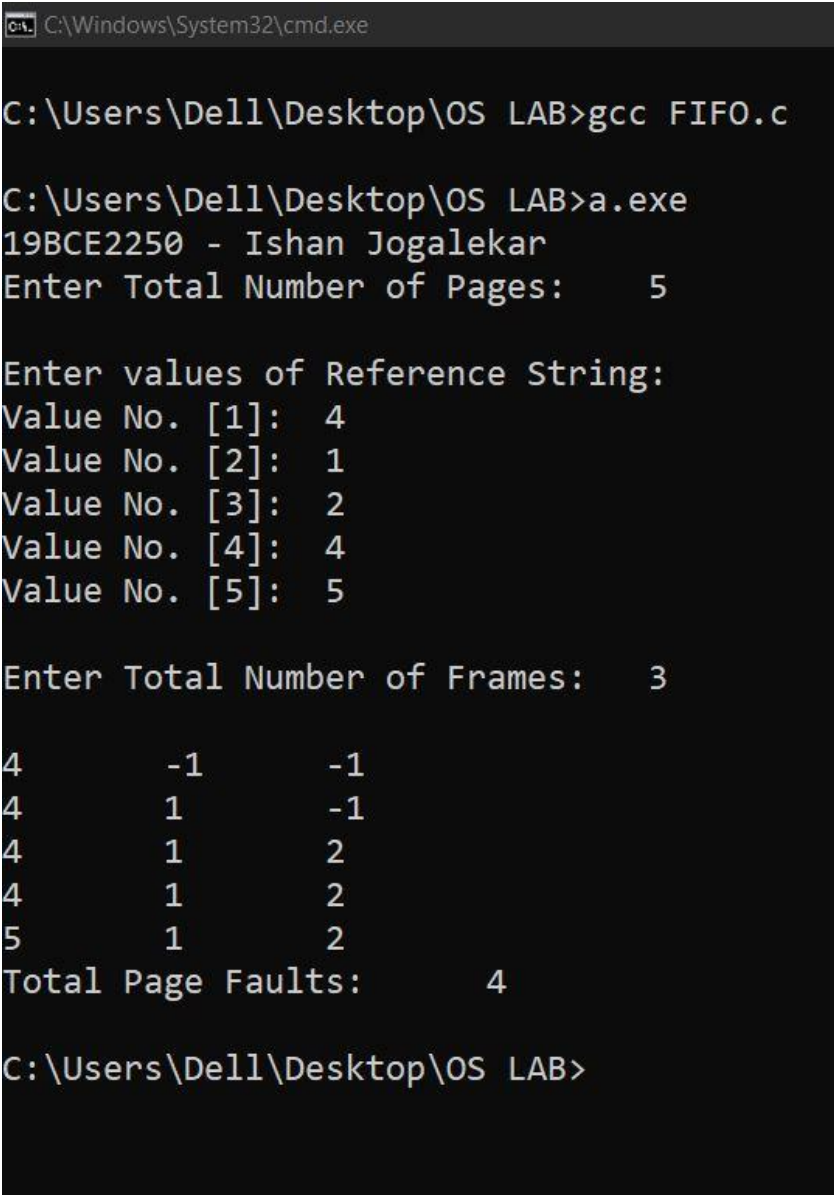
        printf("%d\t", temp[n]);
    }
}

printf("\nTotal Page Faults:\t%d\n", page_faults);

return 0;
}

```

Output :



```

C:\Windows\System32\cmd.exe

C:\Users\De11\Desktop\OS LAB>gcc FIFO.c

C:\Users\De11\Desktop\OS LAB>a.exe
19BCE2250 - Ishan Jogalekar
Enter Total Number of Pages:    5

Enter values of Reference String:
Value No. [1]:  4
Value No. [2]:  1
Value No. [3]:  2
Value No. [4]:  4
Value No. [5]:  5

Enter Total Number of Frames:    3

4      -1      -1
4       1      -1
4       1       2
4       1       2
5       1       2
Total Page Faults:    4

C:\Users\De11\Desktop\OS LAB>

```

## II. LRU :

Ans:

Code –

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

```
int findLRU(int time[], int n){
```

```
    int i, minimum = time[0], pos = 0;
```

```
    for(i = 1; i < n; ++i){
```

```
        if(time[i] < minimum){
```

```
            minimum = time[i];
```

```
            pos = i;
```

```
        }
```

```
    }
```

```
    return pos;
```

```
}
```

```
int main()
```

```
{
```

```
    printf("\n 19BCE2250 - Ishan Jogalekar");
```

```
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0,  
time[10], flag1, flag2, i, j, pos, faults = 0;
```

```
    printf("\nEnter number of frames: ");
```

```
    scanf("%d", &no_of_frames);
```

```
    printf("\nEnter number of pages: ");
```

```
    scanf("%d", &no_of_pages);
```

```
    printf("\nEnter reference string: ");
```

```
for(i = 0; i < no_of_pages; ++i){  
    scanf("%d", &pages[i]);  
}
```

```
for(i = 0; i < no_of_frames; ++i){  
    frames[i] = -1;  
}
```

```
for(i = 0; i < no_of_pages; ++i){  
    flag1 = flag2 = 0;  
    for(j = 0; j < no_of_frames; ++j){  
        if(frames[j] == pages[i]){  
            counter++;  
            time[j] = counter;  
            flag1 = flag2 = 1;  
            break;  
        }  
    }  
}
```

```
if(flag1 == 0){  
    for(j = 0; j < no_of_frames; ++j){  
        if(frames[j] == -1){  
            counter++;  
            faults++;  
            frames[j] = pages[i];  
            time[j] = counter;  
            flag2 = 1;  
        }  
    }  
}
```

```
        break;
    }
}
}
if(flag2 == 0){
    pos = findLRU(time, no_of_frames);
    counter++;
    faults++;
    frames[pos] = pages[i];
    time[pos] = counter;
}
printf("\n");
for(j = 0; j < no_of_frames; ++j){
    printf("%d\t", frames[j]);
}
}
printf("\nTotal Page Faults = %d", faults);
return 0;
}
```

Output:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop\OS LAB>gcc LRU.c

C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter number of frames: 3

Enter number of pages: 5

Enter reference string: 4 1 2 4 5

4      -1      -1
4       1      -1
4       1       2
4       1       2
4       5       2
Total Page Faults = 4
C:\Users\Dell\Desktop\OS LAB>
```

III) OPT :-

Ans:

Code –

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

```
int main()
```

```
{
```

```
    printf("\n19BCE2250 - Ishan Jogalekar");
```

```
    int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1,
flag2, flag3, i, j, k, pos, max, faults = 0;
```

```
    printf("\nEnter number of frames: ");
```

```
    scanf("%d", &no_of_frames);
```

```
    printf("\nEnter number of pages: ");
```

```
scanf("%d", &no_of_pages);
```

```
printf("\nEnter page reference string: ");
```

```
for(i = 0; i < no_of_pages; ++i){
```

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

```
}
```

```
for(i = 0; i < no_of_frames; ++i){
```

```
    frames[i] = -1;
```

```
}
```

```
for(i = 0; i < no_of_pages; ++i){
```

```
    flag1 = flag2 = 0;
```

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

```
        if(frames[j] == pages[i]){
```

```
            flag1 = flag2 = 1;
```

```
            break;
```

```
        }
```

```
    }
```

```
if(flag1 == 0){
```

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

```
        if(frames[j] == -1){
```



```
        faults++;
        frames[j] = pages[i];
        flag2 = 1;
        break;
    }
}

if(flag2 == 0){
    flag3 = 0;

    for(j = 0; j < no_of_frames; ++j){
        temp[j] = -1;

        for(k = i + 1; k < no_of_pages; ++k){
            if(frames[j] == pages[k]){
                temp[j] = k;
                break;
            }
        }
    }

    for(j = 0; j < no_of_frames; ++j){
        if(temp[j] == -1){
            pos = j;
```

```
flag3 = 1;
break;
}
}
```

```
if(flag3 ==0){
max = temp[0];
pos = 0;
```

```
for(j = 1; j < no_of_frames; ++j){
if(temp[j] > max){
max = temp[j];
pos = j;
}
}
}
```

```
frames[pos] = pages[i];
faults++;
}
```

```
printf("\n");
```

```
for(j = 0; j < no_of_frames; ++j){
printf("%d\t", frames[j]);
}
```

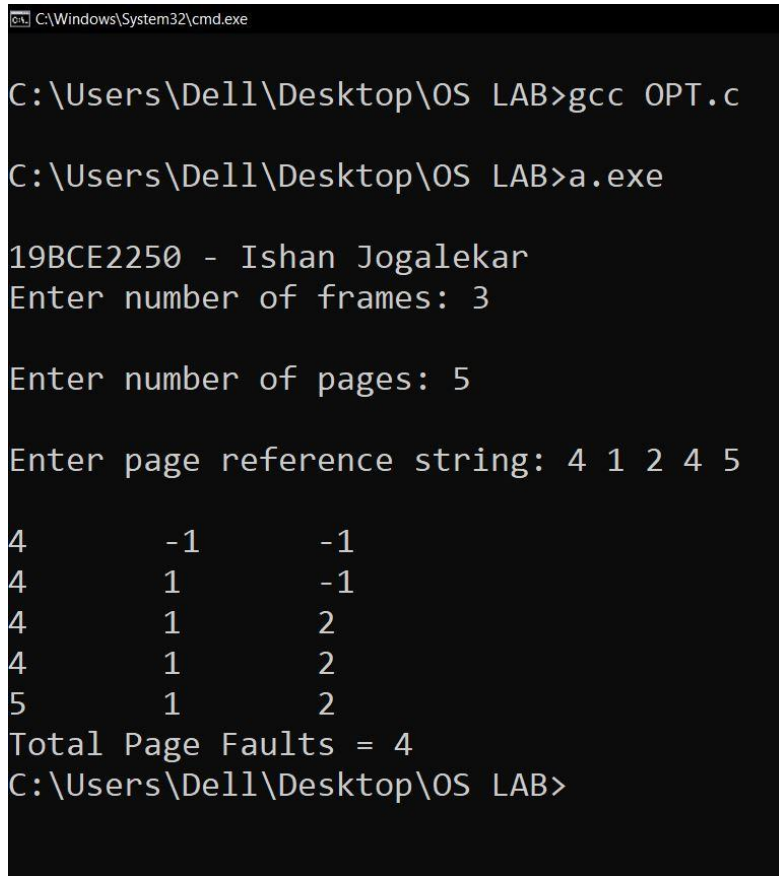
```
}
```

```
printf("\nTotal Page Faults = %d", faults);
```

```
return 0;
```

```
}
```

Output :



```
C:\Windows\System32\cmd.exe

C:\Users\Dell\Desktop\OS LAB>gcc OPT.c

C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter number of frames: 3

Enter number of pages: 5

Enter page reference string: 4 1 2 4 5

4      -1      -1
4       1      -1
4       1       2
4       1       2
5       1       2
Total Page Faults = 4
C:\Users\Dell\Desktop\OS LAB>
```

(c) Write a program that implements the FIFO, LRU, and optimal pager replacement algorithms. First, generate a random page-reference string where page numbers range from 0 to 9. Apply the random page reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithms so that the number of page frames can vary from 1 to 7. Assume that demand paging is used.

I) FIFO :

Ans:

Code –

```
#include <bits/stdc++.h>
#include <string>
#include <iostream>
using namespace std;
int pageFaults (int pages[], int n, int capacity)
{
    unordered_set <int> s;
    queue <int> indexes;
    int page_faults = 0;
    for(int i = 0; i < n; i++)
    {
        if(s.size () < capacity) {
            if(s.find (pages[i]) == s.end ()) {
                s.insert (pages[i]);
                page_faults++;
                indexes.push (pages[i]);
            }
        }
    }
}
```

```

    }
}
else {
    if(s.find (pages[i]) == s.end ()) {
        int val = indexes.front ();
        indexes.pop ();
        s.erase (val);
        s.insert (pages[i]);
        indexes.push (pages[i]);
        page_faults++;
    }
}
}
return page_faults;
}

```

```

int main() {
    cout << "Algorithm = FIFO \n";
    cout<<"\n 19BCE2250 - Ishan Jogalekar";
    int pages[] = { 3, 2, 6, 3, 3, 1, 7, 3, 4, 2, 1, 0, 2};
    int n = sizeof(pages) / sizeof(pages[0]);
    int capacity = 7;

    cout << "\nReferece Pages = ";
    cout << "{";
    for(int i = 0; i < n; i++) {
        cout << pages[i]; cout << ", ";
    }
}

```

```

    }

    cout << "}";

    cout << "\nNumber of Page Faults: ";

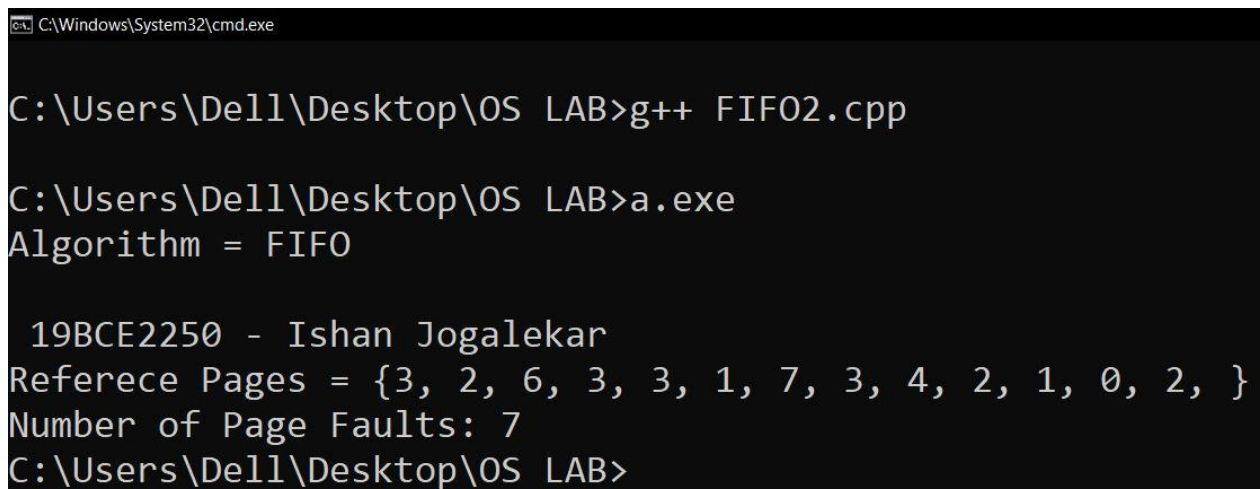
    cout << pageFaults (pages, n, capacity);

    return 0;

}

```

Output :



```

C:\Windows\System32\cmd.exe

C:\Users\Dell\Desktop\OS LAB>g++ FIFO2.cpp

C:\Users\Dell\Desktop\OS LAB>a.exe
Algorithm = FIFO

19BCE2250 - Ishan Jogalekar
Referece Pages = {3, 2, 6, 3, 3, 1, 7, 3, 4, 2, 1, 0, 2, }
Number of Page Faults: 7
C:\Users\Dell\Desktop\OS LAB>

```

II) OPT :

Ans:

Code –

```

#include <bits/stdc++.h>

using namespace std;

bool search(int key, vector<int>& fr)
{
    for (int i = 0; i < fr.size(); i++)
        if (fr[i] == key)

```

```

        return true;
    return false;
}

int predict(int pg[], vector<int>& fr, int pn, int index)
{

    int res = -1, farthest = index;
    for (int i = 0; i < fr.size(); i++) {
        int j;
        for (j = index; j < pn; j++) {
            if (fr[i] == pg[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
    }

    if (j == pn)
        return i;
}

```

```

        return (res == -1) ? 0 : res;
    }

void optimalPage(int pg[], int pn, int fn)
{
    vector<int> fr;
    int hit = 0;
    for (int i = 0; i < pn; i++) {
        if (search(pg[i], fr)) {
            hit++;
            continue;
        }
        if (fr.size() < fn)
            fr.push_back(pg[i]);
        else {
            int j = predict(pg, fr, pn, i + 1);
            fr[j] = pg[i];
        }
    }

    cout << "\nNo. of hits = " << hit << endl;
    cout << "\nNo. of misses = " << pn - hit << endl;
}

// Driver Function
int main()

```

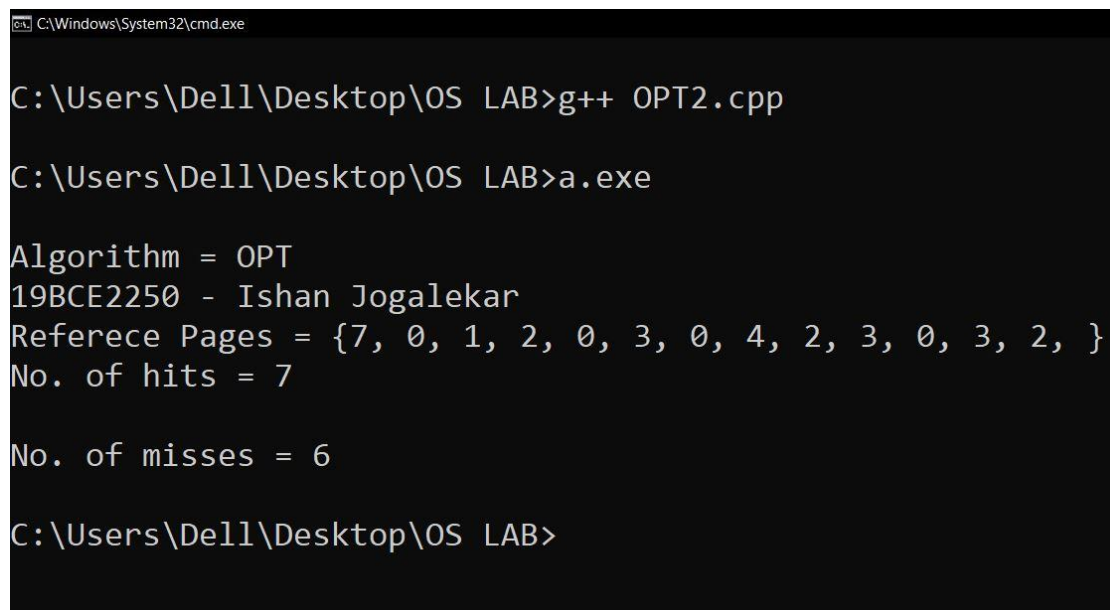


```

{
    cout << "\nAlgorithm = OPT";
    cout<< "\n19BCE2250 - Ishan Jogalekar";
    int pg[] = { 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 };
    int n = sizeof(pg) / sizeof(pg[0]);
    cout << "\nReferece Pages = ";
    cout << "{";
    for(int i = 0; i < n; i++) {
        cout << pg[i];
        cout << ", ";
    }
    cout << "}";
    int fn = 4;
    optimalPage(pg, n, fn);
    return 0;
}

```

Output :



```

C:\Windows\System32\cmd.exe

C:\Users\Dell\Desktop\OS LAB>g++ OPT2.cpp

C:\Users\Dell\Desktop\OS LAB>a.exe

Algorithm = OPT
19BCE2250 - Ishan Jogalekar
Referece Pages = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, }
No. of hits = 7

No. of misses = 6

C:\Users\Dell\Desktop\OS LAB>

```

III) LRU :

Ans :

Code –

```
#include<bits/stdc++.h>

using namespace std;

int pageFaults(int pages[], int n, int capacity)
{
    unordered_set<int> s;
    unordered_map<int, int> indexes;
    int page_faults = 0;
    for (int i=0; i<n; i++)
    {
        if (s.size() < capacity)
        {
            if (s.find(pages[i])==s.end())
            {
                s.insert(pages[i]);
                page_faults++;
            }
            indexes[pages[i]] = i;
        }
        else
        {
            if (s.find(pages[i]) == s.end())
```

```

        {
            int lru = INT_MAX, val;
            for (auto it=s.begin(); it!=s.end(); it++)
            {
                if (indexes[*it] < lru)
                {
                    lru = indexes[*it];
                    val = *it;
                }
            }
            s.erase(val);
            s.insert(pages[i]);
            page_faults++;
        }
        indexes[pages[i]] = i;
    }
}

return page_faults;
}

```

```

int main()
{
    cout << "\nAlgorithm = LRU ";
    cout<<"\n 19BCE2250 - Ishan Jogalekar";
}

```

```

    int pg[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};

    int n = sizeof(pg)/sizeof(pg[0]);

    cout << "\nReferece Pages = ";

    cout << "{";

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

        cout << pg[i];

        cout << ", ";

    }

    cout << "}";

    int capacity = 4;

    cout << "\nPage Faults : ";

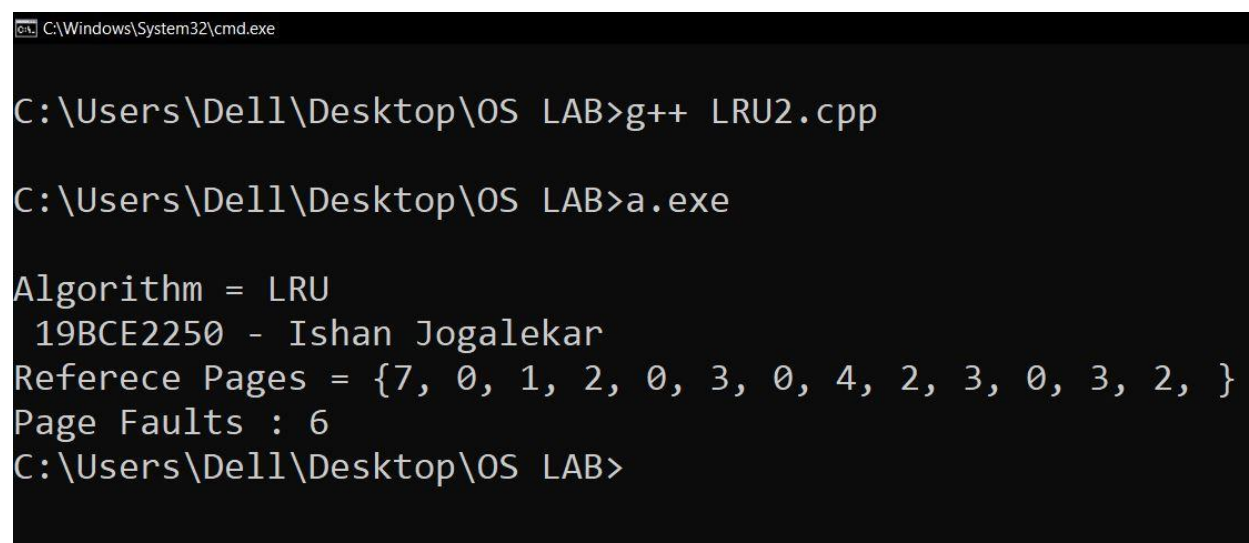
    cout << pageFaults(pg, n, capacity);

    return 0;

}

```

Output :



```

C:\Windows\System32\cmd.exe

C:\Users\Dell\Desktop\OS LAB>g++ LRU2.cpp

C:\Users\Dell\Desktop\OS LAB>a.exe

Algorithm = LRU
19BCE2250 - Ishan Jogalekar
Referece Pages = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, }
Page Faults : 6
C:\Users\Dell\Desktop\OS LAB>

```

## **File system and Disk Management**

(a) Implement the following Disk scheduling algorithms:

1) SSTF :

Ans:

Code –

```
#include<stdio.h>

struct head
{
    int num;
    int flag;
};

int main()
{
    printf("\n 19BCE2250 - Ishan Jogalekar");
    struct head h[33];
    int array_1[33], array_2[33];
    int count = 0, j, x, limit, minimum, location, disk_head, sum = 0;
    printf("\nEnter total number of locations:\t");
    scanf("%d", &limit);
    printf("\nEnter position of disk head:\t");
    scanf("%d", &disk_head);
    printf("\nEnter elements of disk head queue\n");
    while(count < limit)
    {
        scanf("%d", &h[count].num);
```

```

        h[count].flag = 0;
        count++;
    }
    for(count = 0; count < limit; count++)
    {
        x = 0;
        minimum = 0;
        location = 0;
        for(j = 0; j < limit; j++)
        {
            if(h[j].flag == 0)
            {
                if(x == 0)
                {
                    array_1[j] = disk_head - h[j].num;
                    if(array_1[j] < 0)
                    {
                        array_1[j] = h[j].num - disk_head;
                    }
                    minimum = array_1[j];
                    location = j;
                    x++;
                }
                else
                {

```

```

        array_1[j] = disk_head - h[j].num;
        if(array_1[j] < 0)
        {
            array_1[j] = h[j].num - disk_head;
        }
    }
    if(minimum > array_1[j])
    {
        minimum = array_1[j];
        location = j;
    }
}

h[location].flag = 1;
array_2[count] = h[location].num - disk_head;
if(array_2[count] < 0)
{
    array_2[count] = disk_head - h[location].num;
}

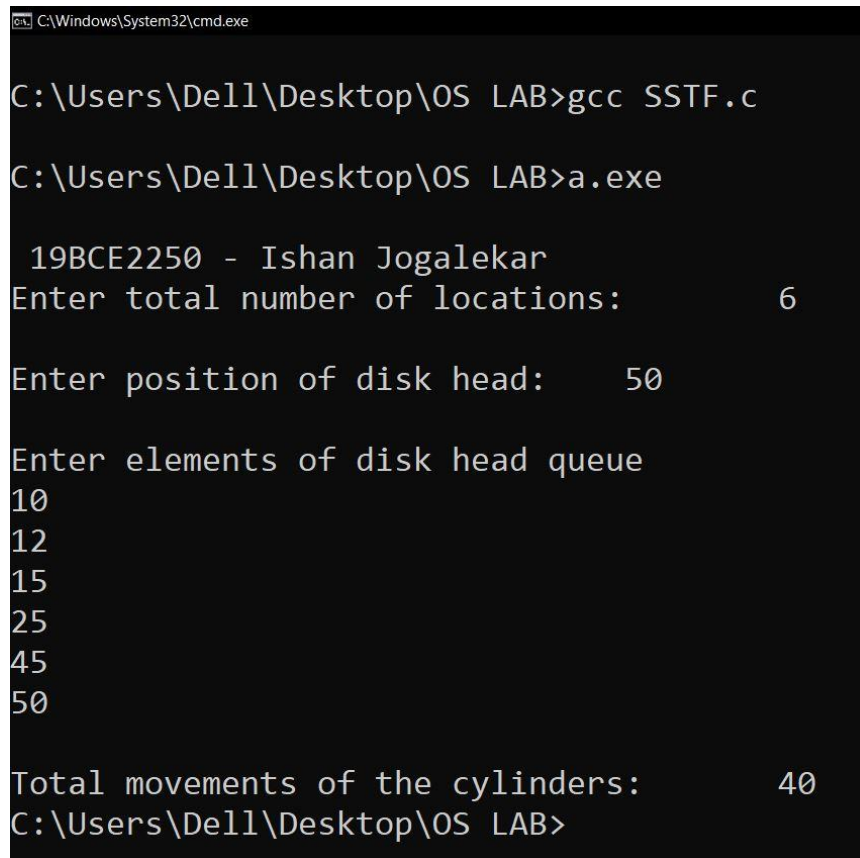
disk_head = h[location].num;
}

count = 0;
while(count < limit)
{
    sum = sum + array_2[count];

```

```
        count++;  
    }  
    printf("\nTotal movements of the cylinders:\t%d", sum);  
    return 0;  
}
```

Output :



```
C:\Windows\System32\cmd.exe  
  
C:\Users\Dell\Desktop\OS LAB>gcc SSTF.c  
  
C:\Users\Dell\Desktop\OS LAB>a.exe  
  
19BCE2250 - Ishan Jogalekar  
Enter total number of locations:          6  
  
Enter position of disk head:      50  
  
Enter elements of disk head queue  
10  
12  
15  
25  
45  
50  
  
Total movements of the cylinders:          40  
C:\Users\Dell\Desktop\OS LAB>
```



II) SCAN :

Ans :

Code –

```
#include<stdio.h>

int main(){

    printf("\n19BCE2250 - Ishan Jogalekar");

    int i,j,sum=0,n;

    int d[20];

    int disk,temp,max,dloc;

    printf("\nEnter number of location:");

    scanf("%d",&n);

    printf("\nEnter position of head:");

    scanf("%d",&disk);

    printf("\nEnter elements of disk queue:");

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

    {

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

    }

    d[n]=disk;

    n=n+1;

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

    {

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

        {

            if(d[i]>d[j])
```

```

        {
            temp=d[i];
            d[i]=d[j];
            d[j]=temp;
        }
    }
}
max=d[n];
for(i=0;i<n;i++)
{
    if(disk==d[i]) { dloc=i; break; }
}
for(i=dloc;i>=0;i--)
{
    printf("%d -->",d[i]);
}
printf("0 -->");
for(i=dloc+1;i<n;i++)
{
    printf("%d-->",d[i]);
}
sum=disk+max;
printf("\nmovement of total cylinders %d",sum);
return 0;
}

```

Output :

```
C:\Windows\System32\cmd.exe

C:\Users\De11\Desktop\OS LAB>gcc SCAN.c

C:\Users\De11\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter number of location:5

Enter position of head:50

enter elements of disk queue:25
35
45
50
55
50 -->45 -->35 -->25 -->0 -->50-->55-->
movement of total cylinders 4200434
C:\Users\De11\Desktop\OS LAB>
```

III) CSCAN :

Ans:

Code –

```
#include<bits/stdc++.h>

using namespace std;

int main(){

    cout<<"\n19BCE2250 - Ishan Jogalekar";

    int i,j,k,n,m,sum=0,x,y,h;

    cout<<"\n Enter the size of disk:";

    cin>>m;

    cout<<"\nEnter number of requests :";
```

```

cin>>n;
cout<<"\nEnter the requests :";
vector <int> a(n),b;
for(i=0;i<n;i++){
    cin>>a[i];
}
for(i=0;i<n;i++){
    if(a[i]>m){
        cout<<"Error, Unknown position " <<a[i]<<"\n";
        return 0;
    }
}
cout<<"\nEnter the head position : ";
cin>>h;
int temp=h;
a.push_back(h);
a.push_back(m);
a.push_back(0);
sort(a.begin(),a.end());
for(i=0;i<a.size();i++){
    if(h==a[i])
        break;
}
k=i;
if(k<n/2){

```

```

    for(i=k;i<a.size();i++){
        b.push_back(a[i]);
    }
    for(i=0;i<=k-1;i++){
        b.push_back(a[i]);
    }
}
else{
    for(i=k;i>=0;i--){
        b.push_back(a[i]);
    }
    for(i=a.size()-1;i>=k+1;i--){
        b.push_back(a[i]);
    }
}
temp=b[0];
cout<<temp;
for(i=1;i<b.size();i++){
    cout<<" -> "<<b[i];
    sum+=abs(b[i]-temp);
    temp=b[i];
}
cout<<"\n";
cout<<"Total head movements = "<< sum;
return 0;}

```

Output :

```
C:\Windows\System32\cmd.exe

C:\Users\Dell\Desktop\OS LAB>g++ CSCAN.cpp

C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter the size of disk:200

Enter number of requests :5

Enter the requests :23
45
56
67
78

Enter the head position : 50
50 -> 45 -> 23 -> 0 -> 200 -> 78 -> 67 -> 56
Total head movements = 394
C:\Users\Dell\Desktop\OS LAB>
```

IV) FCFS :

Ans:

Code –

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

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

```
#include <unistd.h>
```

```
#include <pthread.h>
```

```
int main(int argc, char const *argv[])
```

```
{
```

```
    printf("\n19BCE2250 - Ishan Jogalekar");
```

```
    int range, queue_size, cur_pos, cur_seek_time, total_seek_time = 0;
```

```
printf("\nEnter the size of disk : ");
scanf("%d", &range);
printf("\nEnter the number requests : ");
scanf("%d", &queue_size);

int req_queue[queue_size];
printf("\nEnter the requests : ");
for (int i = 0; i < queue_size; i++)
{
    scanf("%d ", &req_queue[i]);
}
int hold;
scanf("%d", &hold);
printf("\nEnter the head position : ");
scanf("%d", &cur_pos);
for (int i = 0; i < queue_size; i++)
{
    cur_seek_time = abs(req_queue[i] - cur_pos);
    printf("\nDisk head moves from %d to %d with seek time %d",
cur_pos, req_queue[i], cur_seek_time);
    cur_pos = req_queue[i];
    total_seek_time += cur_seek_time;
}
printf("\nTotal Seek Time is : %d", total_seek_time);
return 0;}
```

Output :

```
C:\Windows\System32\cmd.exe

C:\Users\Dell\Desktop\OS LAB>gcc FCFS.c

C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Enter the size of disk : 200

Enter the number requests : 5

Enter the requests : 25
35
40
45
65
75

Enter the head position : 50

Disk head moves from 50 to 25 with seek time 25
Disk head moves from 25 to 35 with seek time 10
Disk head moves from 35 to 40 with seek time 5
Disk head moves from 40 to 45 with seek time 5
Disk head moves from 45 to 65 with seek time 20
Total Seek Time is : 65
C:\Users\Dell\Desktop\OS LAB>_
```



(b) Consider a file of size 1 MB. The size of a disk block is 512Bytes. Assume any number of available free blocks in the disk contiguously or non-contiguously. Implement the following algorithms to perform file allocation. Determine the efficiency of each file allocation strategies.

I) Sequential :

Ans:

Code -

```
#include <stdio.h>
#include <stdlib.h>
void recurse(int files[]){
    int flag = 0, startBlock, len, j, k, ch;
    printf("\nEnter the starting block : ");
    scanf("%d", &startBlock);
    printf("\nEnter the length of files: ");
    scanf("%d", &len);
    for (j=startBlock; j<(startBlock+len); j++){
        if (files[j] == 0)
            flag++;
    }
    if(len == flag){
        for (int k=startBlock; k<(startBlock+len); k++){
            if (files[k] == 0){
                files[k] = 1;
                printf("%d\t%d\n", k, files[k]);
            }
        }
    }
}
```

```

    }
    if (k != (startBlock+len-1))
        printf("\nThe file is allocated to the disk");
    }
else
    printf("\nThe file is not allocated to the disk");

printf("\nEnter more files?\n");
printf("\n1 for YES, 0 for NO: ");
scanf("%d", &ch);
if (ch == 1)
    recurse(files);
else
    exit(0);
return;
}

```

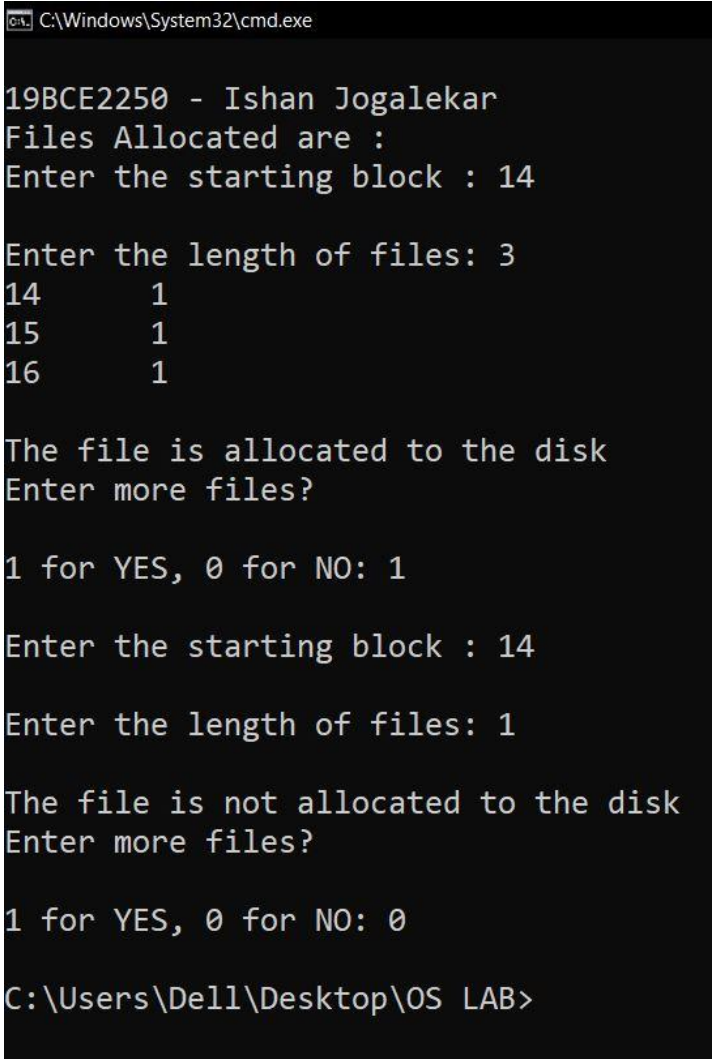
```

int main()
{
    printf("\n19BCE2250 - Ishan Jogalekar");
    int files[50];
    for(int i=0;i<50;i++)
        files[i]=0;
    printf("\nFiles Allocated are :");
    recurse(files);
}

```

```
    return 0;  
}
```

Output :



```
C:\Windows\System32\cmd.exe  
  
19BCE2250 - Ishan Jogalekar  
Files Allocated are :  
Enter the starting block : 14  
  
Enter the length of files: 3  
14      1  
15      1  
16      1  
  
The file is allocated to the disk  
Enter more files?  
  
1 for YES, 0 for NO: 1  
  
Enter the starting block : 14  
  
Enter the length of files: 1  
  
The file is not allocated to the disk  
Enter more files?  
  
1 for YES, 0 for NO: 0  
  
C:\Users\De11\Desktop\OS LAB>
```

II) Index :

Ans:

Code –

```
#include <stdio.h>  
  
#include <stdlib.h>  
  
int files[50], indexBlock[50], indBlock, n;  
  
void recurse1();
```

```
void recurse2();
```

```
void recurse1(){
```

```
    printf("\nEnter the index block: ");
```

```
    scanf("%d", &indBlock);
```

```
    if (files[indBlock] != 1){
```

```
        printf("\nEnter the number of blocks and the number of files  
needed for the index %d on the disk: ", indBlock);
```

```
        scanf("%d", &n);
```

```
    }
```

```
    else{
```

```
        printf("\n%d is already allocated", indBlock);
```

```
        recurse1();
```

```
    }
```

```
    recurse2();
```

```
}
```

```
void recurse2(){
```

```
    int ch;
```

```
    int flag = 0;
```

```
    for (int i=0; i<n; i++){
```

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

```
        if (files[indexBlock[i]] == 0)
```

```
            flag++;
```

```
    }
```

```

if (flag == n){
    for (int j=0; j<n; j++){
        files[indexBlock[j]] = 1;
    }
    printf("\nAllocated");
    printf("\nFile Indexed");
    for (int k=0; k<n; k++){
        printf("\n%d -----> %d : %d", indBlock, indexBlock[k],
files[indexBlock[k]]);
    }
}
else{
    printf("\nFile in the index is already allocated");
    printf("\nEnter another indexed file");
    recurse2();
}

printf("\nDo you want to enter more files?");
printf("\n1 for Yes, Enter 0 for No: ");
scanf("%d", &ch);
if (ch == 1)
    recurse1();
else
    exit(0);
return;
}

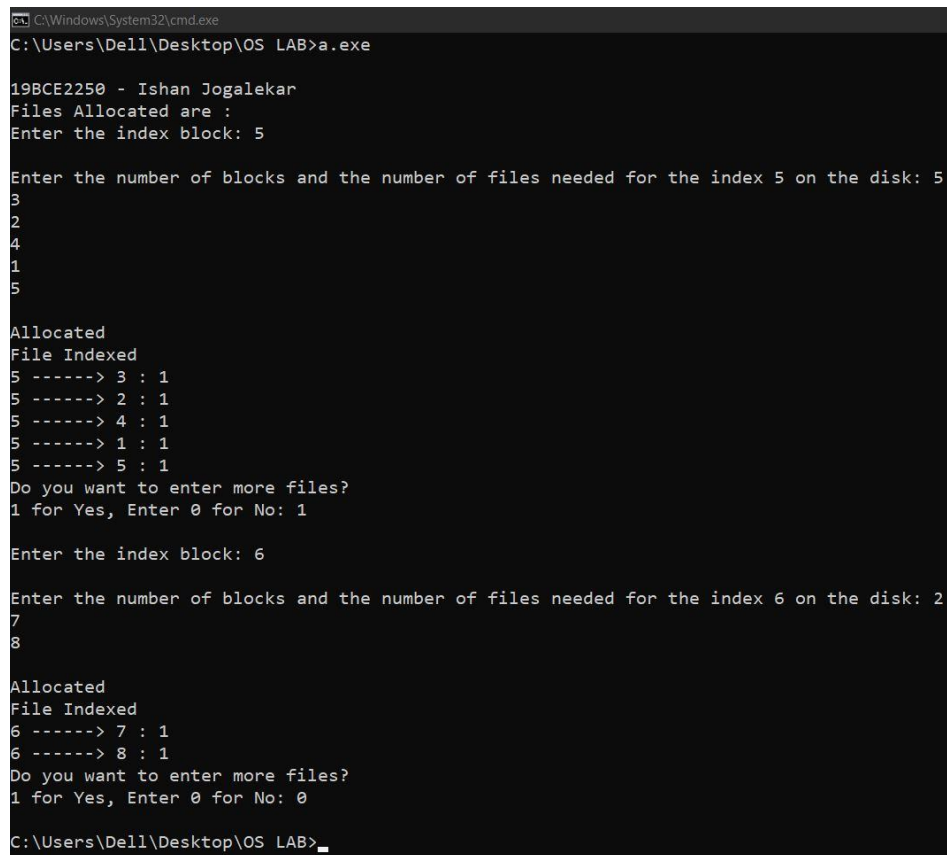
```

```

int main()
{
    printf("\n19BCE2250 - Ishan Jogalekar");
    for(int i=0;i<50;i++)
        files[i]=0;
    printf("\nFiles Allocated are :");
    recurse1();
    return 0;
}

```

Output :



```

C:\Windows\System32\cmd.exe
C:\Users\Dell\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalekar
Files Allocated are :
Enter the index block: 5

Enter the number of blocks and the number of files needed for the index 5 on the disk: 5
3
2
4
1
5

Allocated
File Indexed
5 -----> 3 : 1
5 -----> 2 : 1
5 -----> 4 : 1
5 -----> 1 : 1
5 -----> 5 : 1
Do you want to enter more files?
1 for Yes, Enter 0 for No: 1

Enter the index block: 6

Enter the number of blocks and the number of files needed for the index 6 on the disk: 2
7
8

Allocated
File Indexed
6 -----> 7 : 1
6 -----> 8 : 1
Do you want to enter more files?
1 for Yes, Enter 0 for No: 0

C:\Users\Dell\Desktop\OS LAB>_

```

III) Linked :

Ans:

Code –

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

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

```
void recursivePart(int pages[]){
```

```
    int st, len, k, c, j;
```

```
    printf("\nEnter the index of the starting block: ");
```

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

```
    printf("\nEnter the length: ");
```

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

```
    k = len;
```

```
    if (pages[st] == 0){
```

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

```
            if (pages[j] == 0){
```

```
                pages[j] = 1;
```

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

```
            }
```

```
        else {
```

```
            printf("\nThe block %d is already allocated", j);
```

```
            k++;
```

```
        }
```

```
    }
```

```
}
```

```
else
```

```

        printf("\nThe block %d is already allocated", st);
    printf("\nDo you want to enter more files?");
    printf("\n1 for Yes, Enter 0 for No: ");
    scanf("%d", &c);
    if (c == 1)
        recursivePart(pages);
    else
        exit(0);
    return;
}
int main(){
    printf("\n19BCE2250 - Ishan Jogalkear");
    int pages[50], p, a;
    for (int i = 0; i < 50; i++)
        pages[i] = 0;
    printf("\nEnter the number of blocks already allocated: ");
    scanf("%d", &p);
    printf("\nEnter the blocks already allocated: ");
    for (int i = 0; i < p; i++){
        scanf("%d", &a);
        pages[a] = 1;
    }
    recursivePart(pages);
    return 0;
}

```



Output :

```
C:\Windows\System32\cmd.exe

C:\Users\De11\Desktop\OS LAB>gcc Linked.c

C:\Users\De11\Desktop\OS LAB>a.exe

19BCE2250 - Ishan Jogalkear
Enter the number of blocks already allocated: 3

Enter the blocks already allocated: 2 4 6

Enter the index of the starting block: 2

Enter the length: 3

The block 2 is already allocated
Do you want to enter more files?
1 for Yes, Enter 0 for No: 1

Enter the index of the starting block: 5

Enter the length: 1
5----->1

Do you want to enter more files?
1 for Yes, Enter 0 for No: 0

C:\Users\De11\Desktop\OS LAB>
```