ITE2002-Operating System (Lab)

WINTER SEM 20-21

Assessment-5

Name :Pravin G

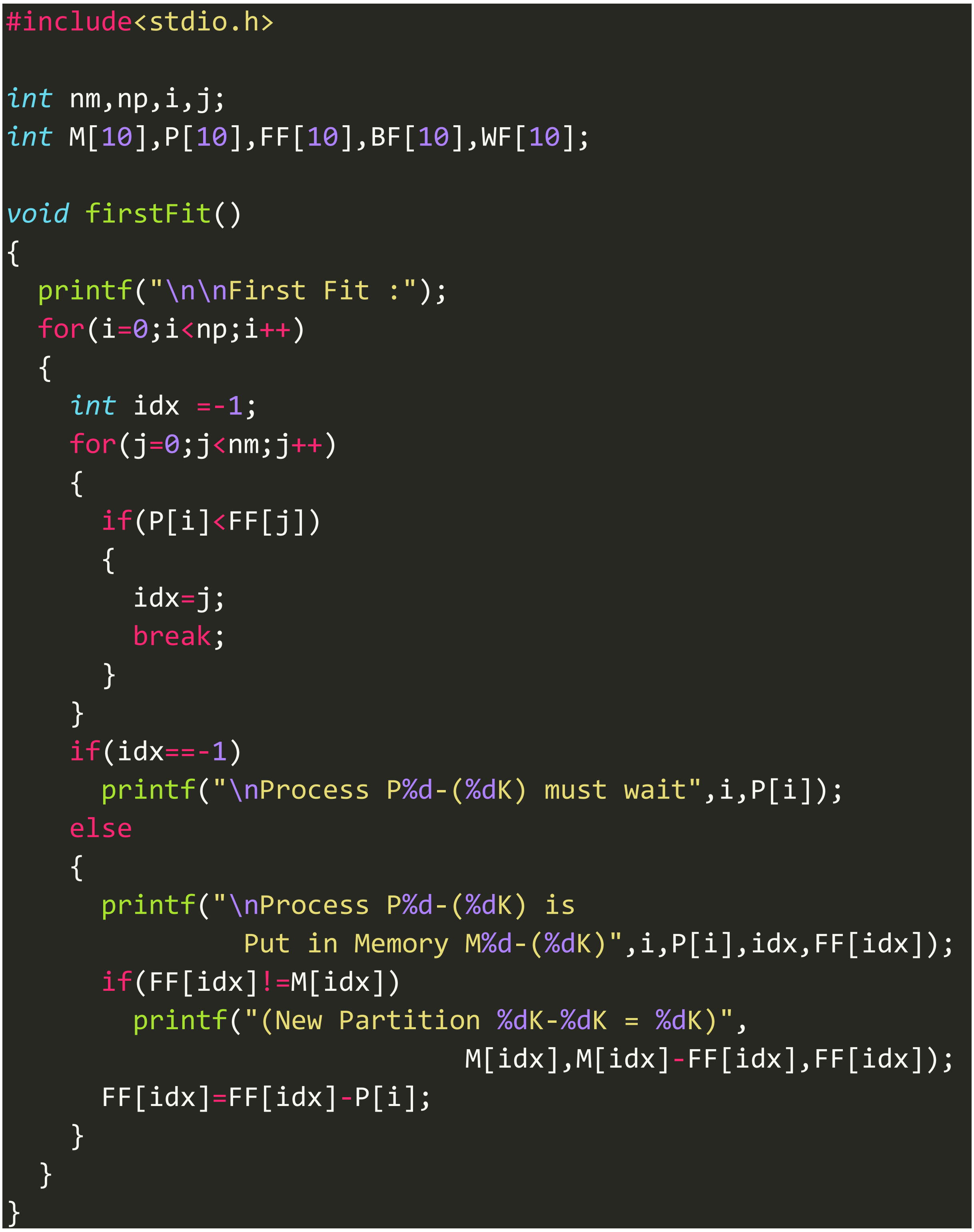
Reg No :19BIT0393

Slot :L41+L42

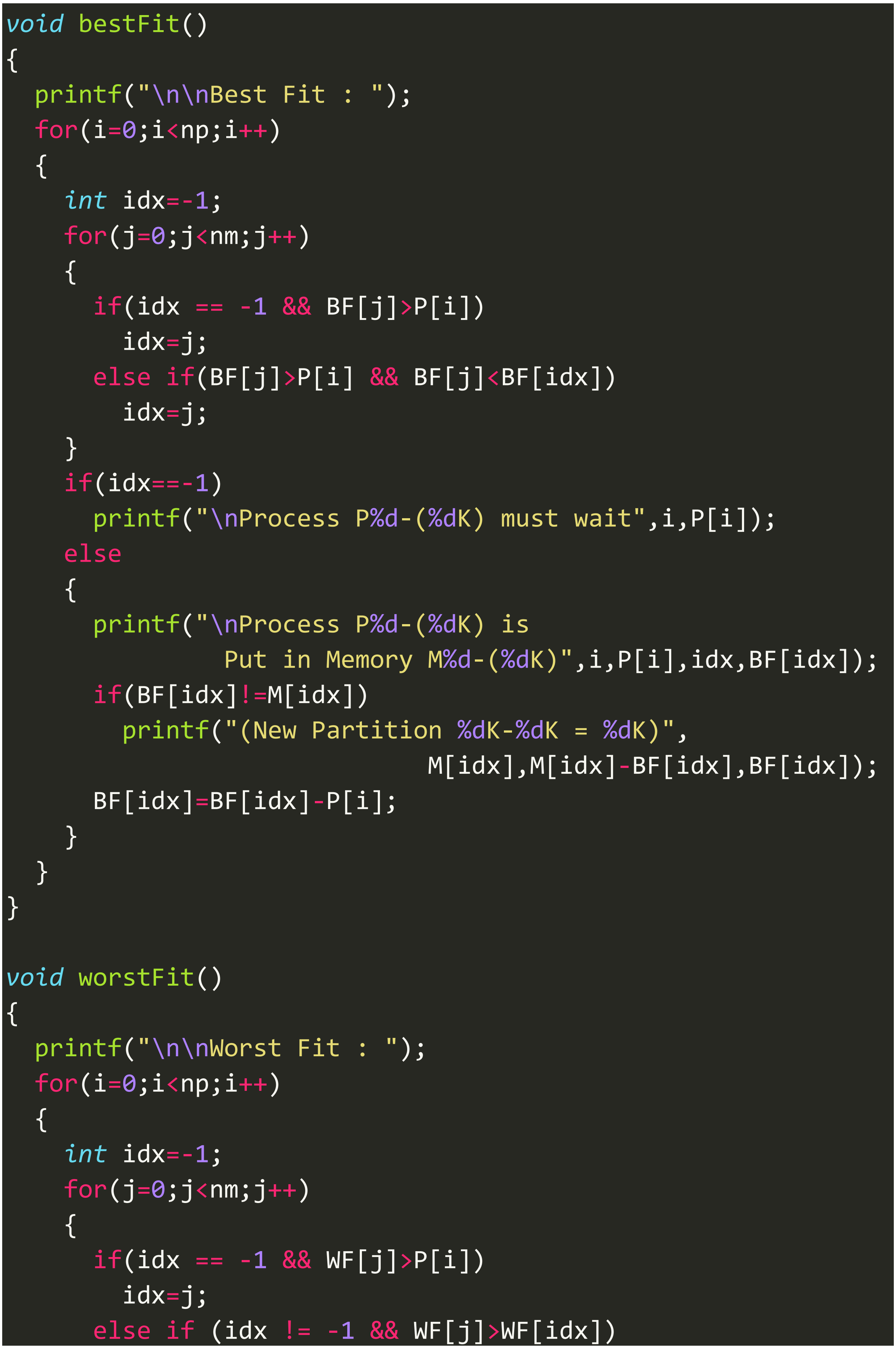
Question 1

Write a program to implement the first fit, best fit, and worst fit algorithm for memory allocation.

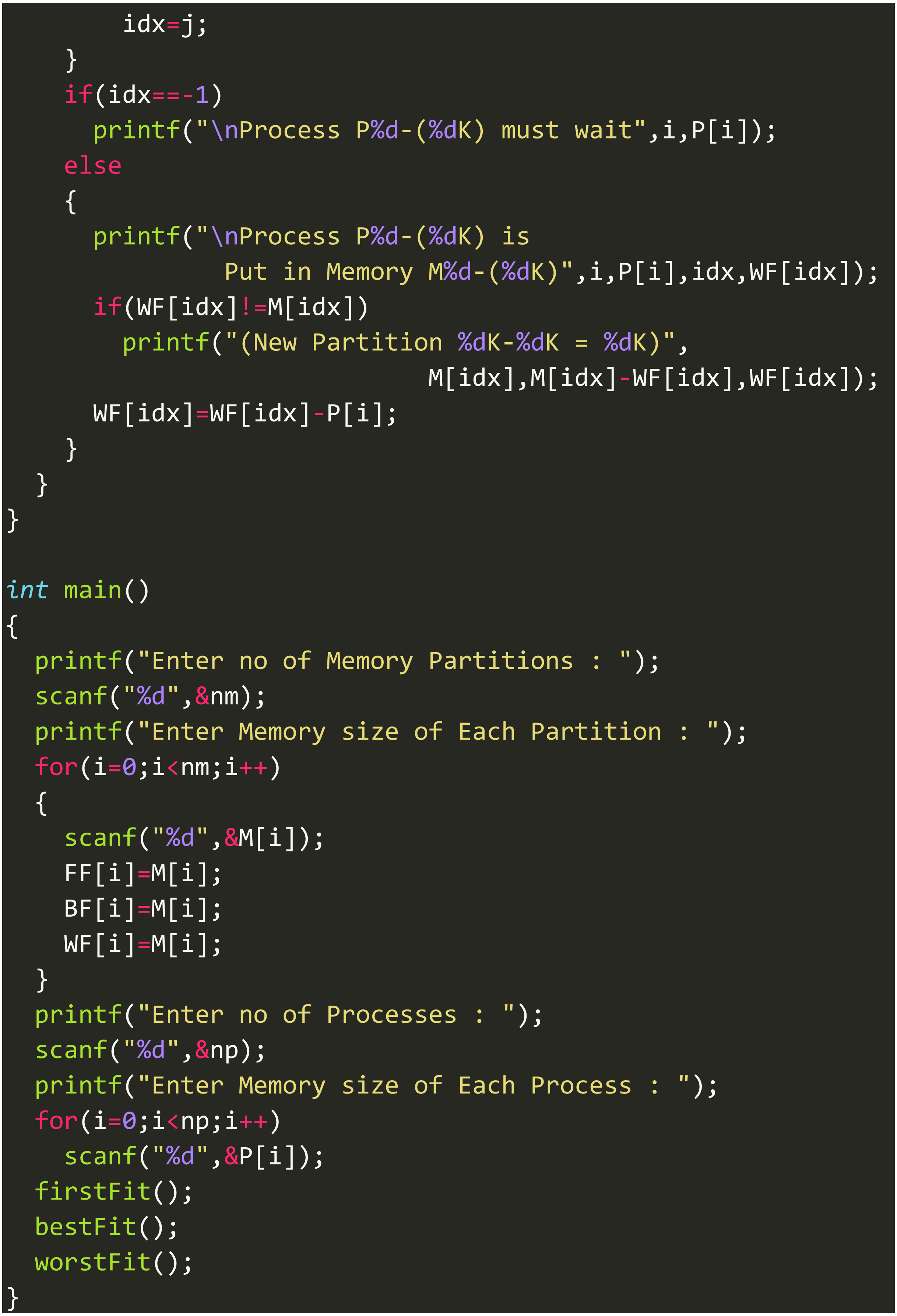
Code:-



```c
#include<stdio.h>

int nm,np,i,j;
int M[10],P[10],FF[10],BF[10],WF[10];

void firstFit()
{
  printf("\n\nFirst Fit :");
  for(i=0;i<np;i++)
  {
    int idx =-1;
    for(j=0;j<nm;j++)
    {
      if(P[i]<FF[j])
      {
        idx=j;
        break;
      }
    }
    if(idx==-1)
      printf("\nProcess P%d-(%dK) must wait",i,P[i]);
    else
    {
      printf("\nProcess P%d-(%dK) is
               Put in Memory M%d-(%dK)",i,P[i],idx,FF[idx]);
      if(FF[idx]!=M[idx])
        printf("(New Partition %dK-%dK = %dK)",
                             M[idx],M[idx]-FF[idx],FF[idx]);
      FF[idx]=FF[idx]-P[i];
    }
  }
}
```

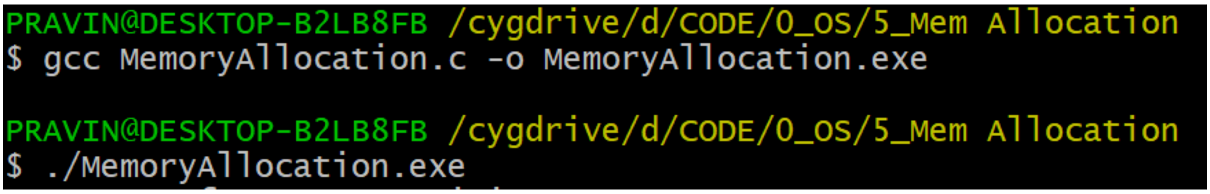


```
void bestFit()
{
  printf("\n\nBest Fit : ");
  for(i=0;i<np;i++)
  {
    int idx=-1;
    for(j=0;j<nm;j++)
    {
      if(idx == -1 && BF[j]>P[i])
        idx=j;
      else if(BF[j]>P[i] && BF[j]<BF[idx])
        idx=j;
    }
    if(idx==-1)
      printf("\nProcess P%d-(%dK) must wait",i,P[i]);
    else
    {
      printf("\nProcess P%d-(%dK) is
               Put in Memory M%d-(%dK)",i,P[i],idx,BF[idx]);
      if(BF[idx]!=M[idx])
        printf("(New Partition %dK-%dK = %dK)",
                             M[idx],M[idx]-BF[idx],BF[idx]);
      BF[idx]=BF[idx]-P[i];
    }
  }
}

void worstFit()
{
  printf("\n\nWorst Fit : ");
  for(i=0;i<np;i++)
  {
    int idx=-1;
    for(j=0;j<nm;j++)
    {
      if(idx == -1 && WF[j]>P[i])
        idx=j;
      else if (idx != -1 && WF[j]>WF[idx])
```



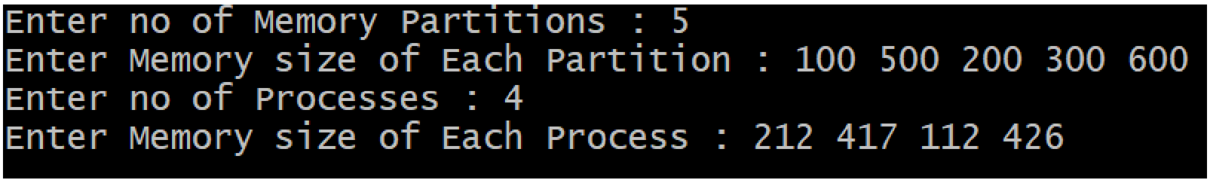
```c
        idx=j;
    }
    if(idx==-1)
      printf("\nProcess P%d-(%dK) must wait",i,P[i]);
    else
    {
      printf("\nProcess P%d-(%dK) is
               Put in Memory M%d-(%dK)",i,P[i],idx,WF[idx]);
      if(WF[idx]!=M[idx])
        printf("(New Partition %dK-%dK = %dK)",
                             M[idx],M[idx]-WF[idx],WF[idx]);
      WF[idx]=WF[idx]-P[i];
    }
  }
}

int main()
{
  printf("Enter no of Memory Partitions : ");
  scanf("%d",&nm);
  printf("Enter Memory size of Each Partition : ");
  for(i=0;i<nm;i++)
  {
    scanf("%d",&M[i]);
    FF[i]=M[i];
    BF[i]=M[i];
    WF[i]=M[i];
  }
  printf("Enter no of Processes : ");
  scanf("%d",&np);
  printf("Enter Memory size of Each Process : ");
  for(i=0;i<np;i++)
    scanf("%d",&P[i]);
  firstFit();
  bestFit();
  worstFit();
}
```

Execution:



```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ gcc MemoryAllocation.c -o MemoryAllocation.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ ./MemoryAllocation.exe
```

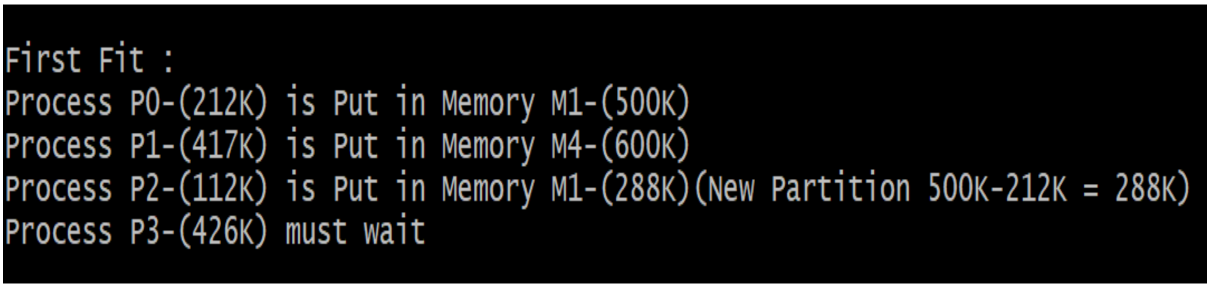
Input:



```
Enter no of Memory Partitions : 5
Enter Memory size of Each Partition : 100 500 200 300 600
Enter no of Processes : 4
Enter Memory size of Each Process : 212 417 112 426
```

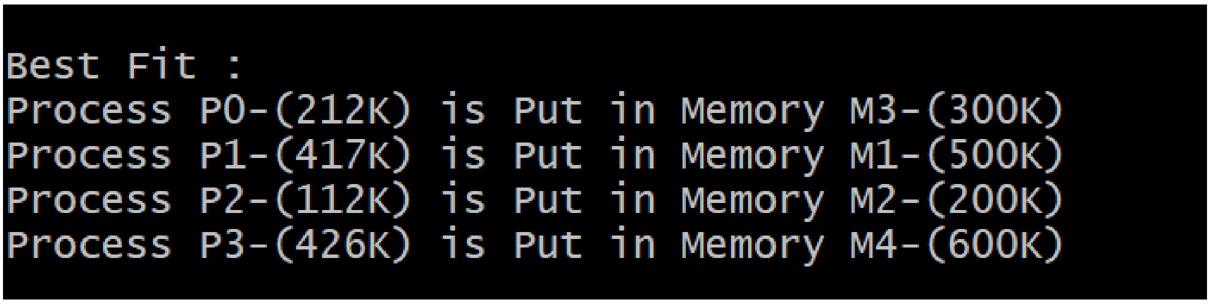
Output:

First Fit



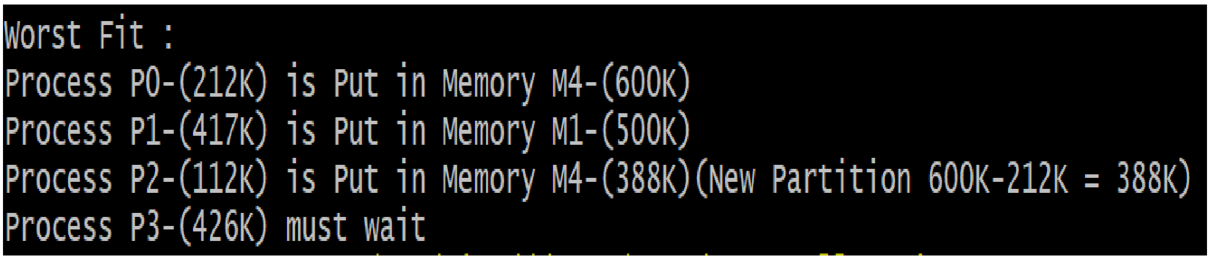
```
First Fit :
Process P0-(212K) is Put in Memory M1-(500K)
Process P1-(417K) is Put in Memory M4-(600K)
Process P2-(112K) is Put in Memory M1-(288K)(New Partition 500K-212K = 288K)
Process P3-(426K) must wait
```

Best Fit



```
Best Fit :
Process P0-(212K) is Put in Memory M3-(300K)
Process P1-(417K) is Put in Memory M1-(500K)
Process P2-(112K) is Put in Memory M2-(200K)
Process P3-(426K) is Put in Memory M4-(600K)
```

Worst Fit



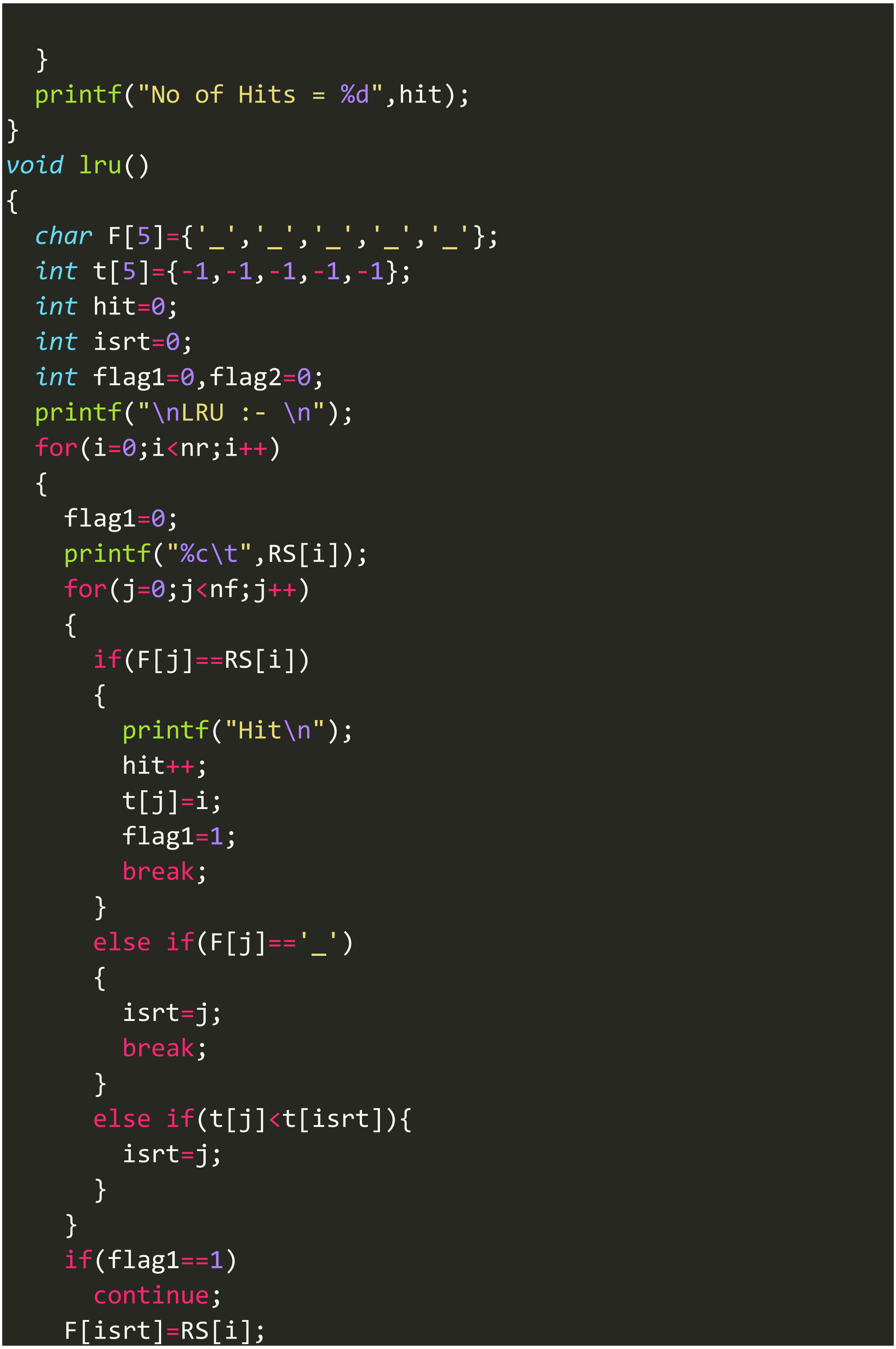
```
Worst Fit :
Process P0-(212K) is Put in Memory M4-(600K)
Process P1-(417K) is Put in Memory M1-(500K)
Process P2-(112K) is Put in Memory M4-(388K)(New Partition 600K-212K = 388K)
Process P3-(426K) must wait
```

Question 2

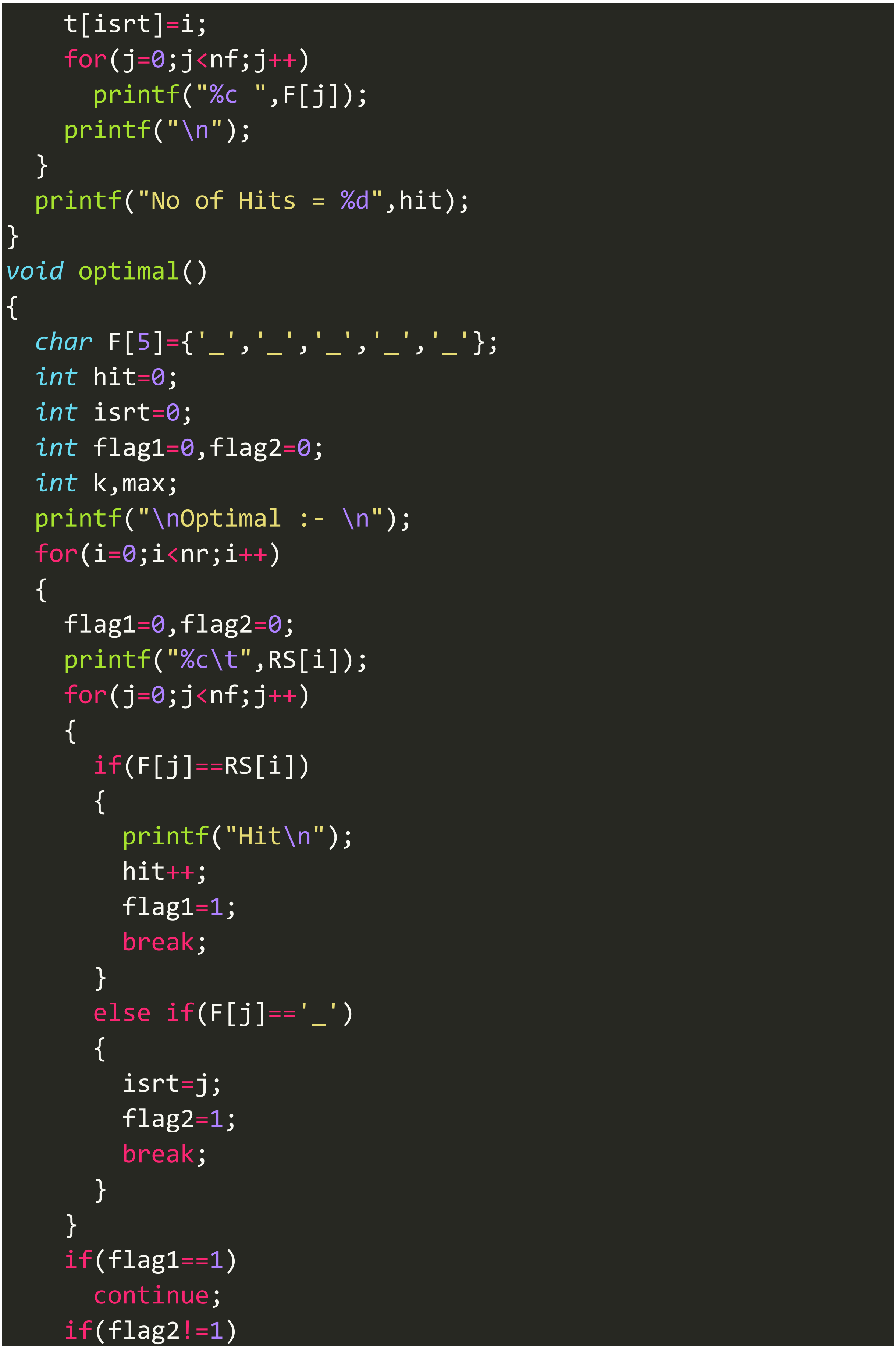
Write a program to implement the page replacement algorithms. a. FIFO b. LRU c. OPTIMAL

Code:-

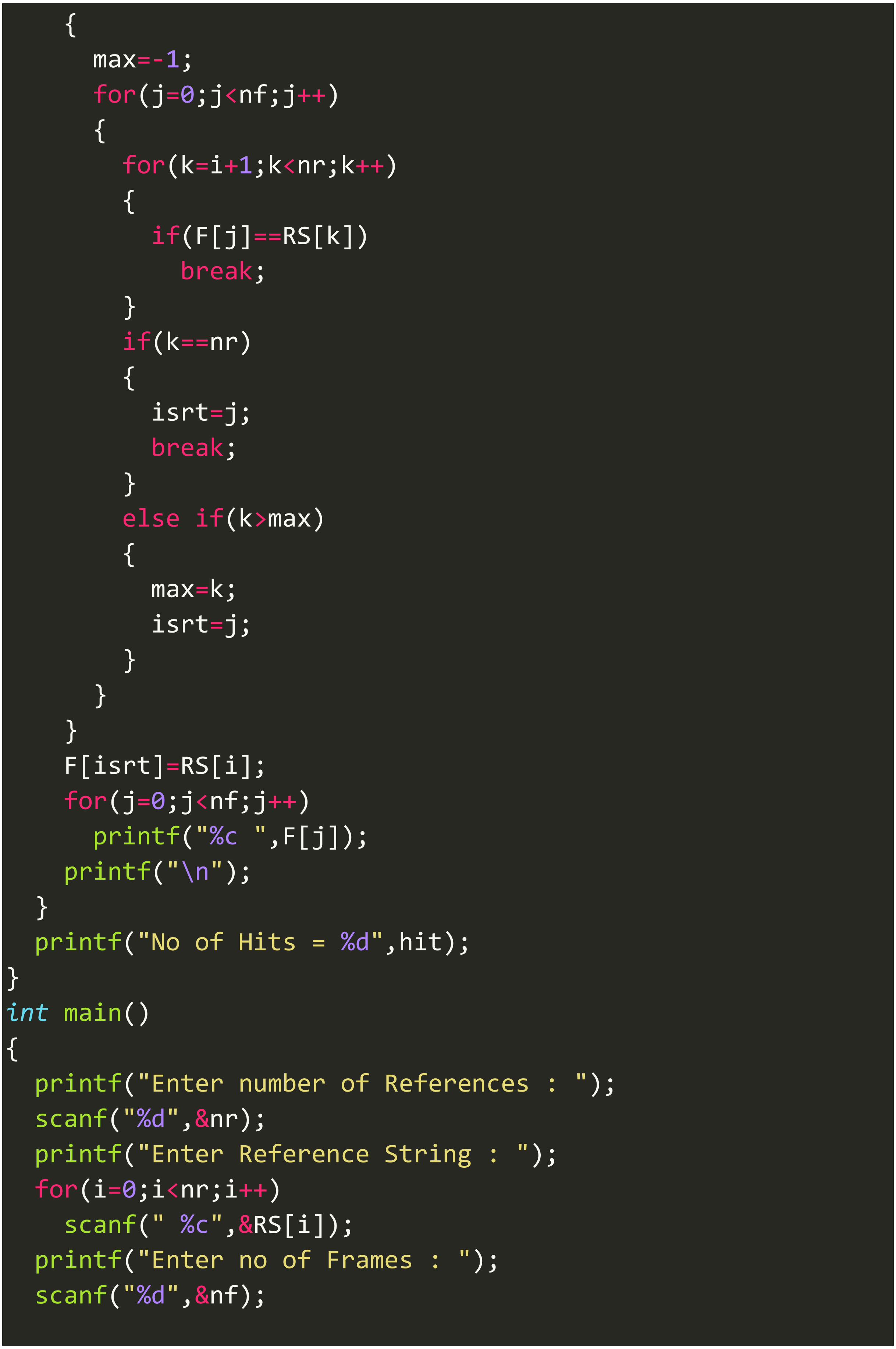
```
#include<stdio.h>
char RS[50];
int nr,nf,i,j;
void fifo()
{
  char F[5]={'_','_','_','_','_'};
  int hit=0;
  int isrt=-1;
  int flag;
  printf("\nFIFO :- \n");
  for(i=0;i<nr;i++)
  {
    flag=0;
    printf("%c\t",RS[i]);
    for(j=0;j<nf;j++)
    {
      if(F[j]==RS[i])
      {
        printf("Hit\n");
        hit++;
        flag=1;
        break;
      }
    }
    if(flag==1)
      continue;
    F[(++isrt)%nf]=RS[i];
    for(j=0;j<nf;j++)
      printf("%c ",F[j]);
    printf("\n");
```



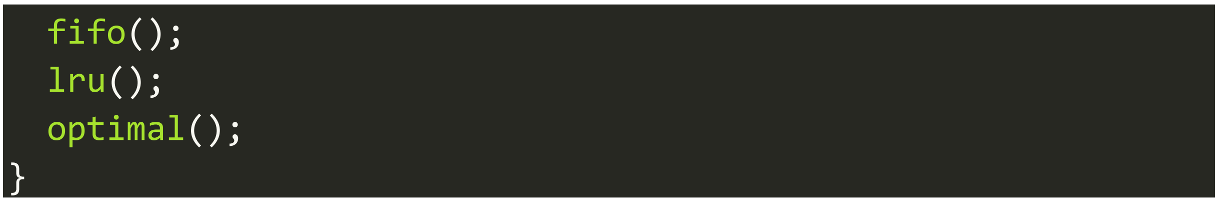
```
  }
  printf("No of Hits = %d",hit);
}
void lru()
{
  char F[5]={'_','_','_','_','_'};
  int t[5]={-1,-1,-1,-1,-1};
  int hit=0;
  int isrt=0;
  int flag1=0,flag2=0;
  printf("\nLRU :- \n");
  for(i=0;i<nr;i++)
  {
    flag1=0;
    printf("%c\t",RS[i]);
    for(j=0;j<nf;j++)
    {
      if(F[j]==RS[i])
      {
        printf("Hit\n");
        hit++;
        t[j]=i;
        flag1=1;
        break;
      }
      else if(F[j]=='_')
      {
        isrt=j;
        break;
      }
      else if(t[j]<t[isrt]){
        isrt=j;
      }
    }
    if(flag1==1)
      continue;
    F[isrt]=RS[i];
```



```
    t[isrt]=i;
    for(j=0;j<nf;j++)
      printf("%c ",F[j]);
    printf("\n");
  }
  printf("No of Hits = %d",hit);
}
void optimal()
{
  char F[5]={'_','_','_','_','_'};
  int hit=0;
  int isrt=0;
  int flag1=0,flag2=0;
  int k,max;
  printf("\nOptimal :- \n");
  for(i=0;i<nr;i++)
  {
    flag1=0,flag2=0;
    printf("%c\t",RS[i]);
    for(j=0;j<nf;j++)
    {
      if(F[j]==RS[i])
      {
        printf("Hit\n");
        hit++;
        flag1=1;
        break;
      }
      else if(F[j]=='_')
      {
        isrt=j;
        flag2=1;
        break;
      }
    }
    if(flag1==1)
      continue;
    if(flag2!=1)
```

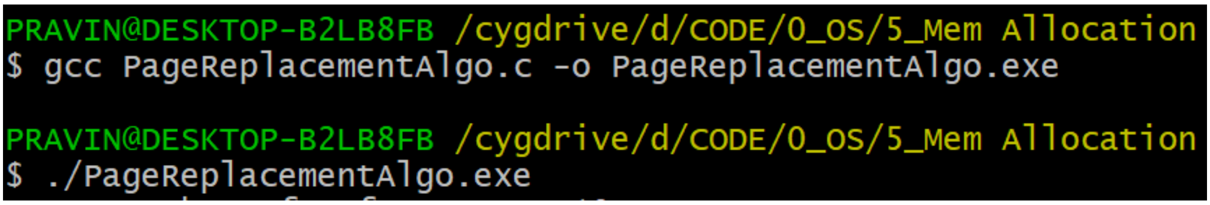


```
    {
      max=-1;
      for(j=0;j<nf;j++)
      {
        for(k=i+1;k<nr;k++)
        {
          if(F[j]==RS[k])
            break;
        }
        if(k==nr)
        {
          isrt=j;
          break;
        }
        else if(k>max)
        {
          max=k;
          isrt=j;
        }
      }
    }
    F[isrt]=RS[i];
    for(j=0;j<nf;j++)
      printf("%c ",F[j]);
    printf("\n");
  }
  printf("No of Hits = %d",hit);
}
int main()
{
  printf("Enter number of References : ");
  scanf("%d",&nr);
  printf("Enter Reference String : ");
  for(i=0;i<nr;i++)
    scanf(" %c",&RS[i]);
  printf("Enter no of Frames : ");
  scanf("%d",&nf);
```



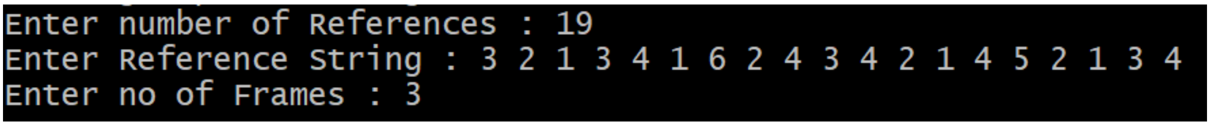
```
  fifo();
  lru();
  optimal();
}
```

Exection:



```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ gcc PageReplacementAlgo.c -o PageReplacementAlgo.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ ./PageReplacementAlgo.exe
```

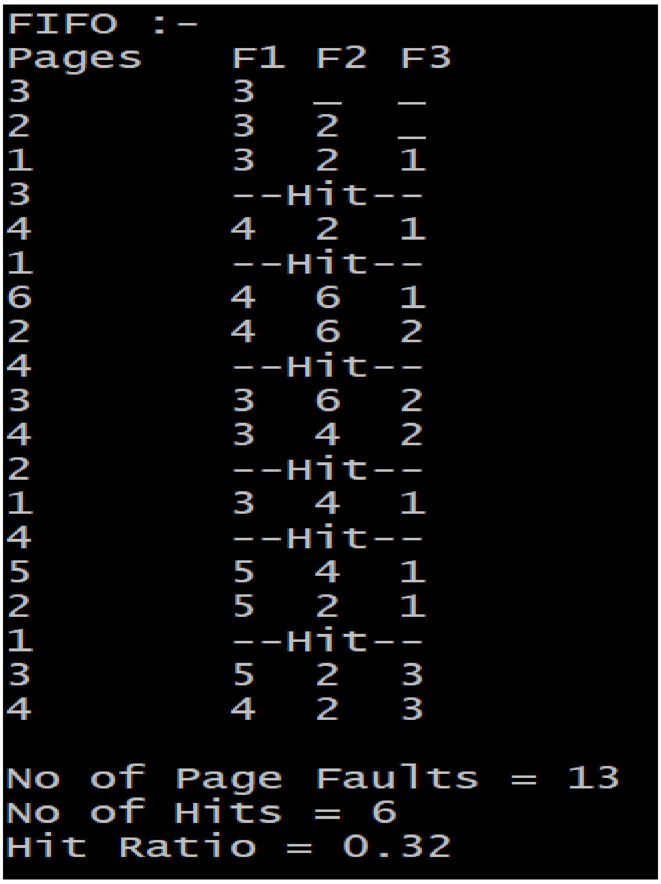
Input:



```
Enter number of References : 19
Enter Reference String : 3 2 1 3 4 1 6 2 4 3 4 2 1 4 5 2 1 3 4
Enter no of Frames : 3
```

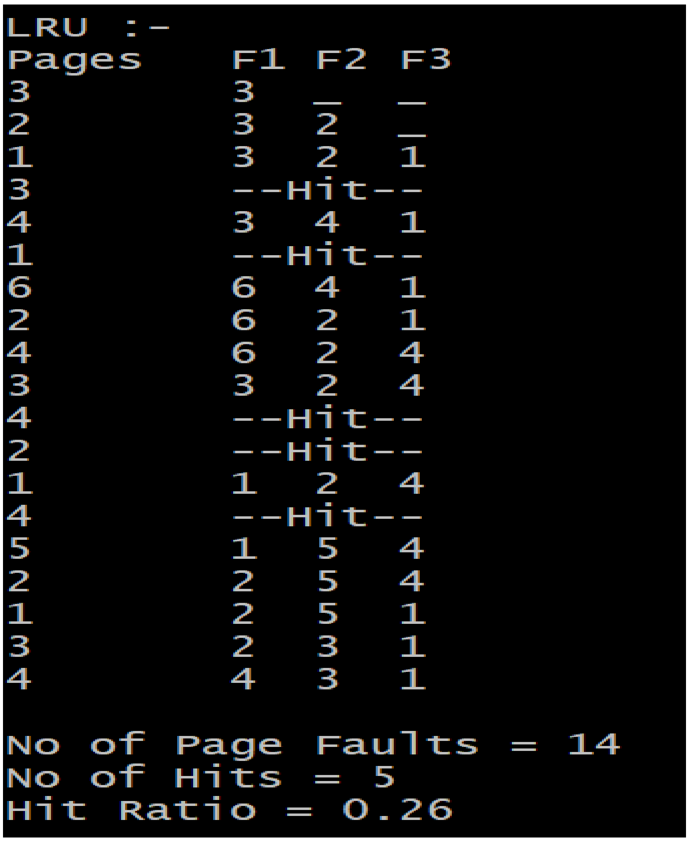
Output:

FIFO



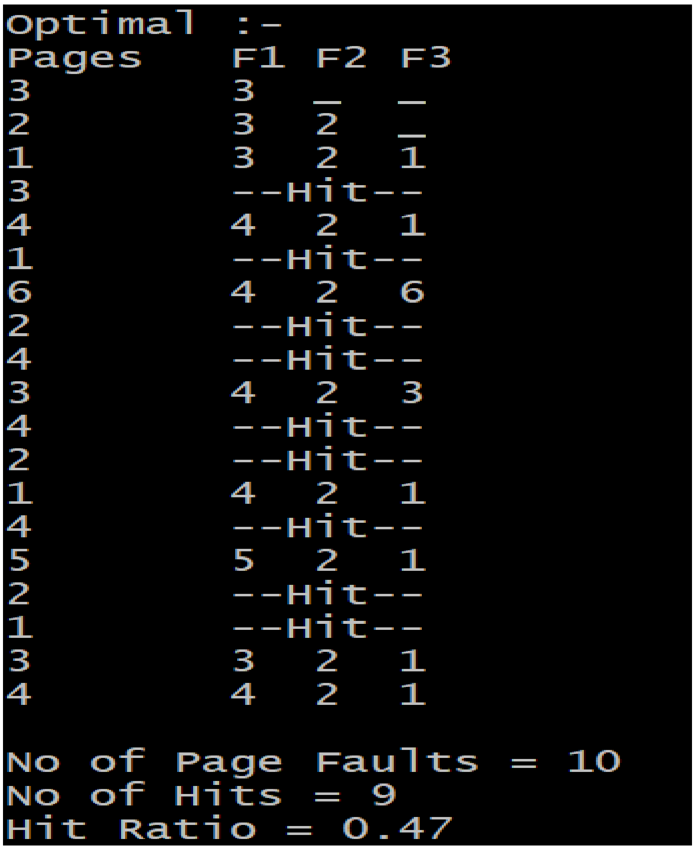
```
FIFO :-
Pages      F1  F2  F3
3          3   _   _
2          3   2   _
1          3   2   1
3          --Hit--
4          4   2   1
1          --Hit--
6          4   6   1
2          4   6   2
4          --Hit--
3          3   6   2
4          3   4   2
2          --Hit--
1          3   4   1
4          --Hit--
5          5   4   1
2          5   2   1
1          --Hit--
3          5   2   3
4          4   2   3

No of Page Faults = 13
No of Hits = 6
Hit Ratio = 0.32
```

LRU



```
LRU :-
Pages   F1 F2 F3
3       3  _  _
2       3  2  _
1       3  2  1
3       --Hit--
4       3  4  1
1       --Hit--
6       6  4  1
2       6  2  1
4       6  2  4
3       3  2  4
4       --Hit--
2       --Hit--
1       1  2  4
4       --Hit--
5       1  5  4
2       2  5  4
1       2  5  1
3       2  3  1
4       4  3  1

No of Page Faults = 14
No of Hits = 5
Hit Ratio = 0.26
```

Optimal



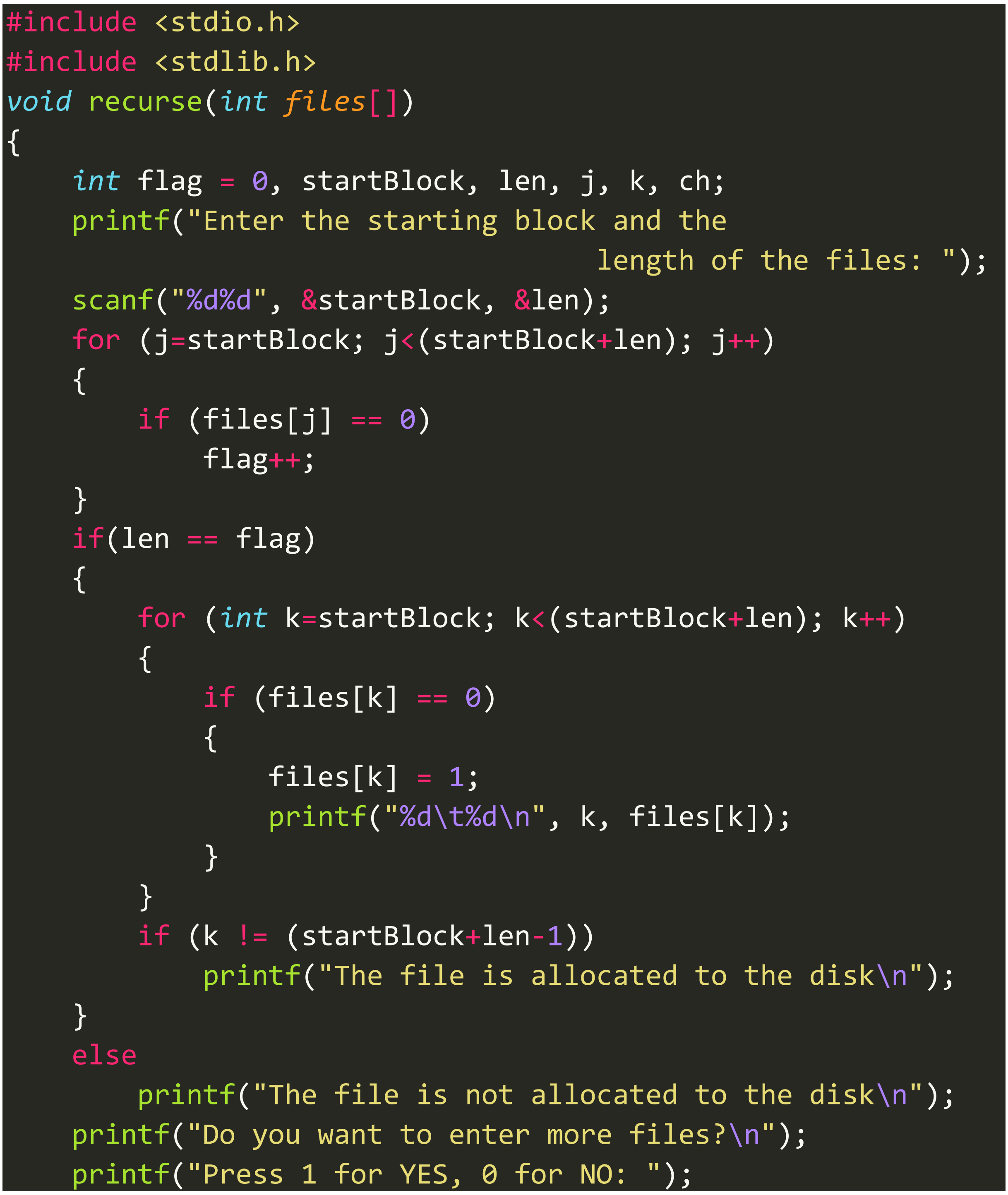
```
Optimal :-
Pages   F1 F2 F3
3       3  _  _
2       3  2  _
1       3  2  1
3       --Hit--
4       4  2  1
1       --Hit--
6       4  2  6
2       --Hit--
4       --Hit--
3       4  2  3
4       --Hit--
2       --Hit--
1       4  2  1
4       --Hit--
5       5  2  1
2       --Hit--
1       --Hit--
3       3  2  1
4       4  2  1

No of Page Faults = 10
No of Hits = 9
Hit Ratio = 0.47
```

Question 3

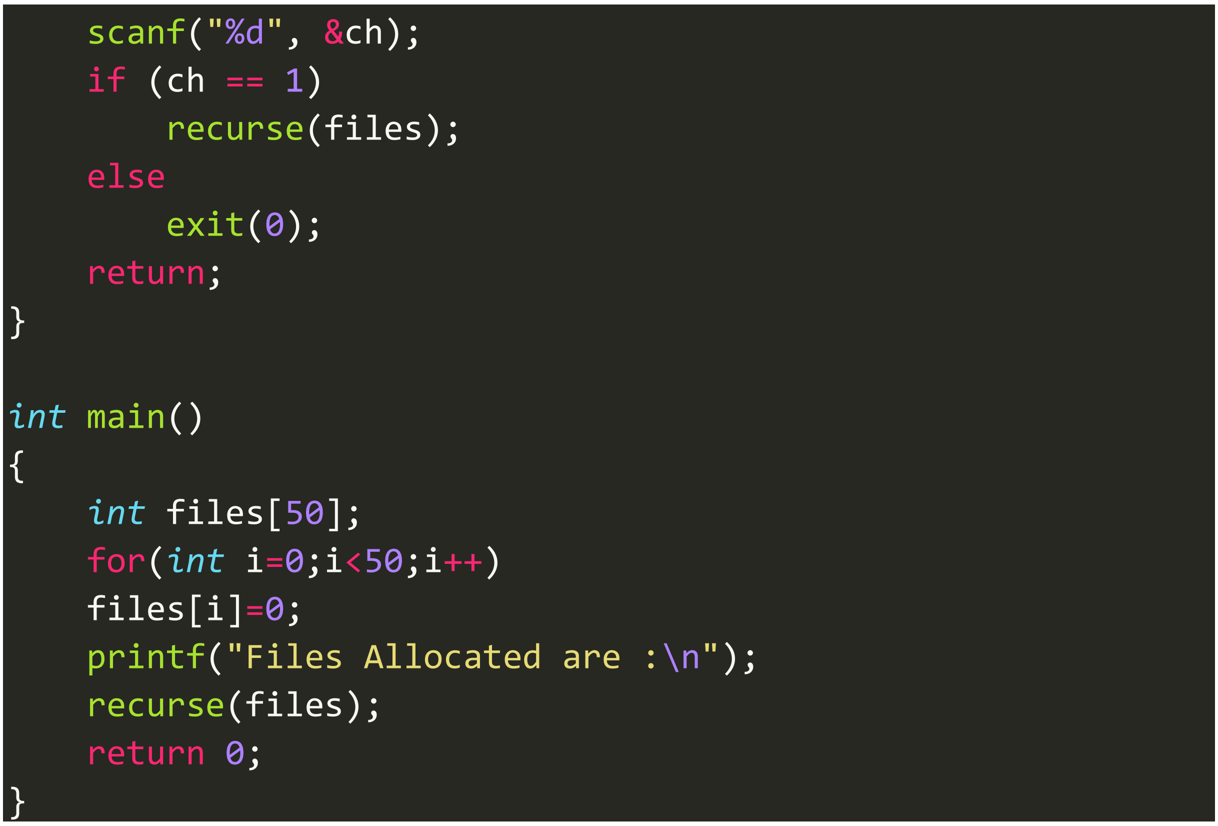
Implement the following algorithms to perform file allocation.

a. Sequential

Code:-

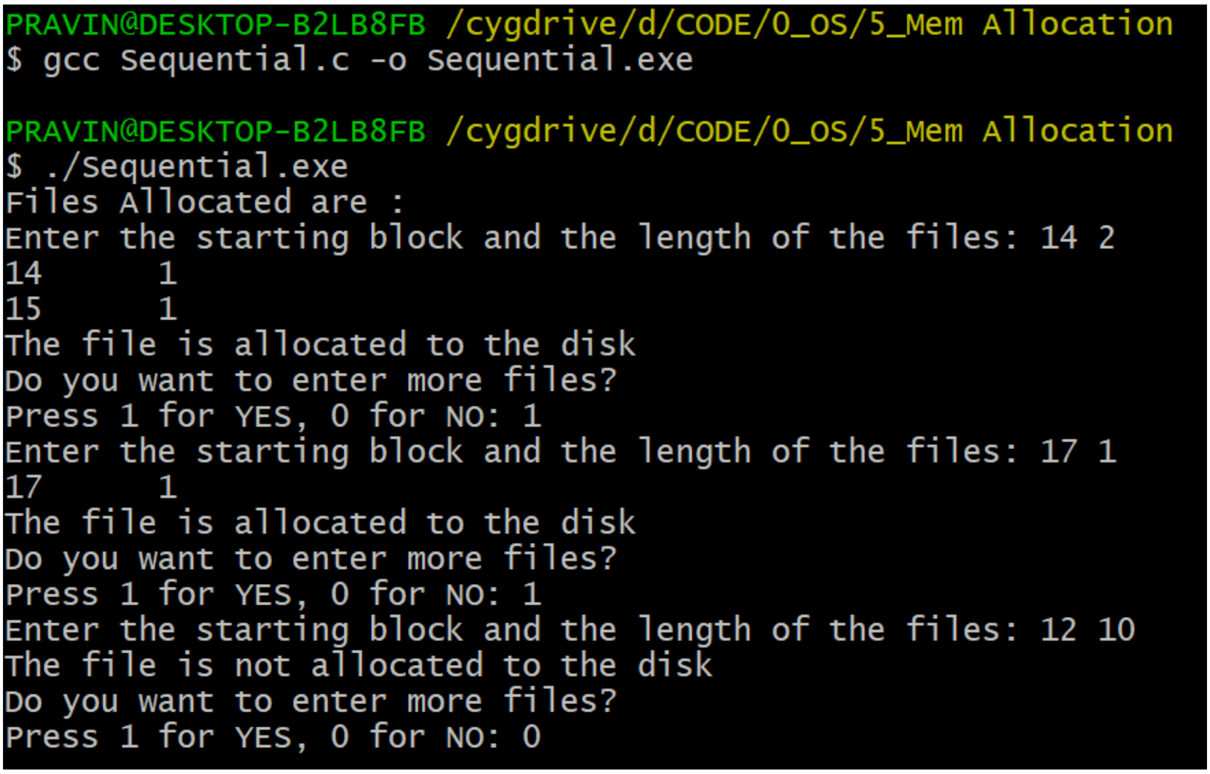


```
#include <stdio.h>
#include <stdlib.h>
void recurse(int files[])
{
    int flag = 0, startBlock, len, j, k, ch;
    printf("Enter the starting block and the
                                    length of the files: ");
    scanf("%d%d", &startBlock, &len);
    for (j=startBlock; j<(startBlock+len); 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("The file is allocated to the disk\n");
    }
    else
        printf("The file is not allocated to the disk\n");
    printf("Do you want to enter more files?\n");
    printf("Press 1 for YES, 0 for NO: ");
```



```c
    scanf("%d", &ch);
    if (ch == 1)
        recurse(files);
    else
        exit(0);
    return;
}

int main()
{
    int files[50];
    for(int i=0;i<50;i++)
    files[i]=0;
    printf("Files Allocated are :\n");
    recurse(files);
    return 0;
}
```

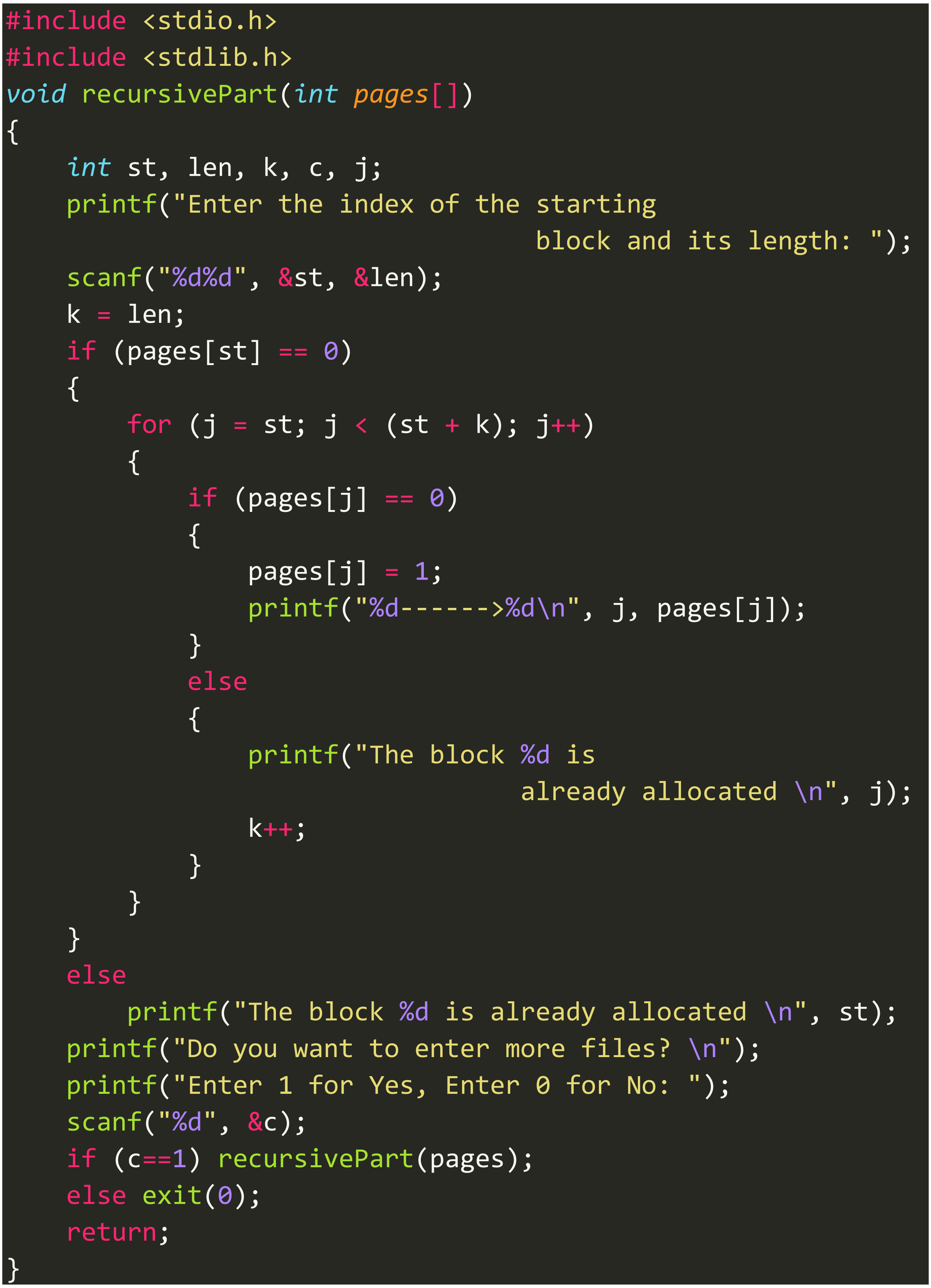
Output:-



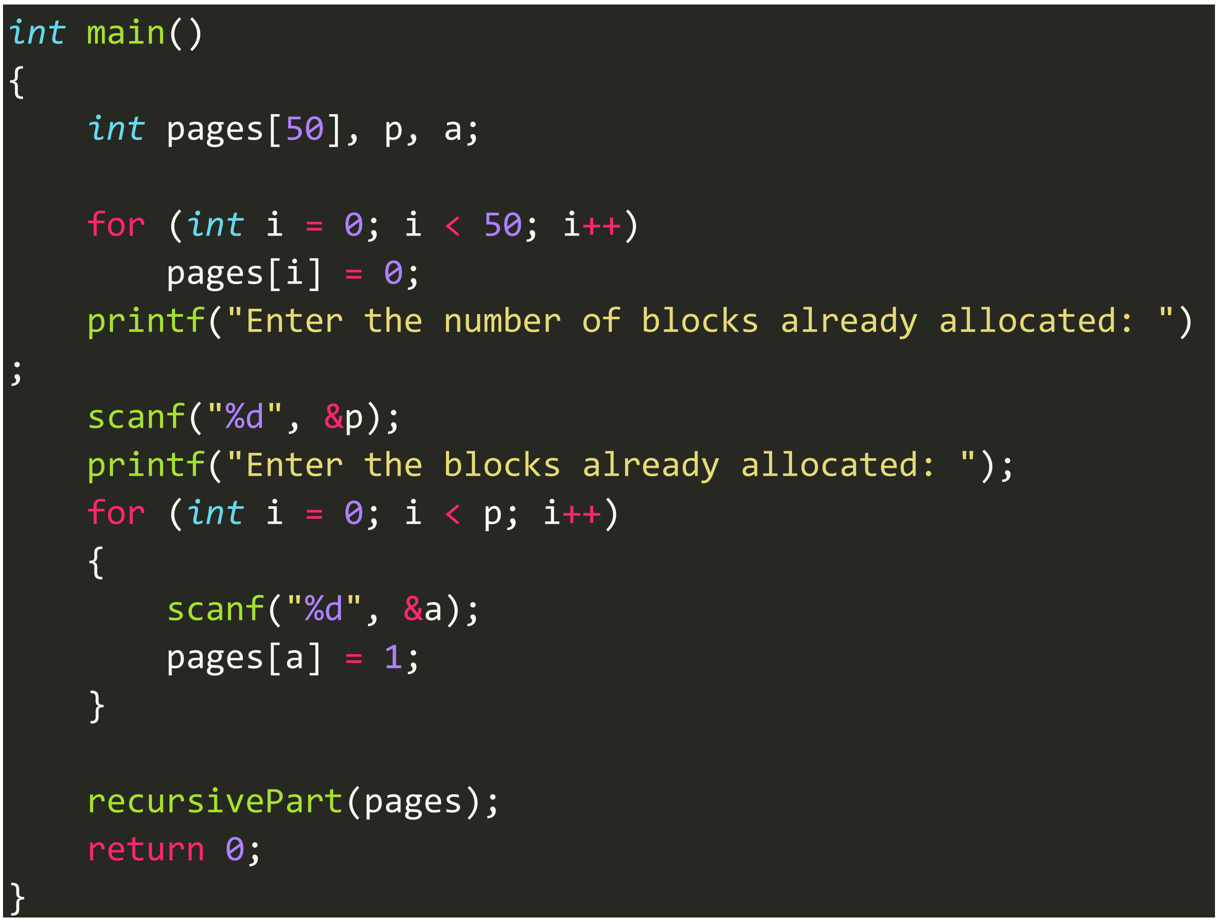
```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ gcc Sequential.c -o Sequential.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ ./Sequential.exe
Files Allocated are :
Enter the starting block and the length of the files: 14 2
14      1
15      1
The file is allocated to the disk
Do you want to enter more files?
Press 1 for YES, 0 for NO: 1
Enter the starting block and the length of the files: 17 1
17      1
The file is allocated to the disk
Do you want to enter more files?
Press 1 for YES, 0 for NO: 1
Enter the starting block and the length of the files: 12 10
The file is not allocated to the disk
Do you want to enter more files?
Press 1 for YES, 0 for NO: 0
```

b. Linked

Code:-

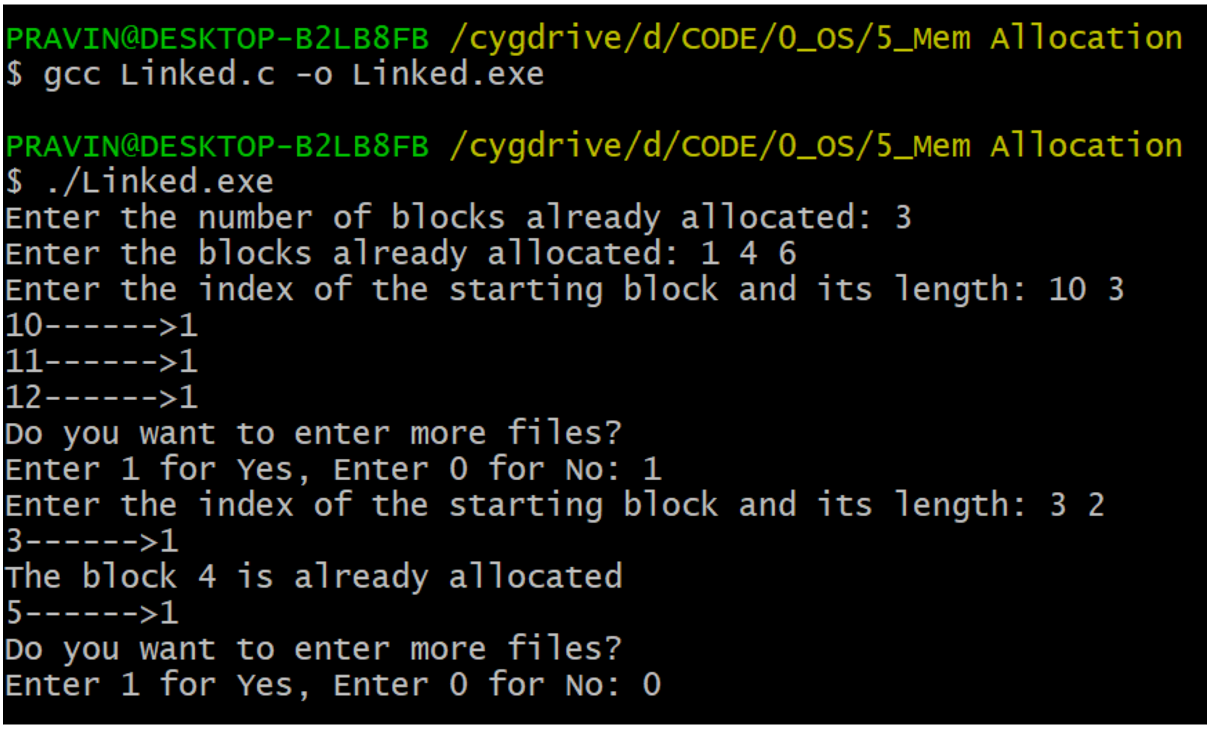


```
#include <stdio.h>
#include <stdlib.h>
void recursivePart(int pages[])
{
    int st, len, k, c, j;
    printf("Enter the index of the starting
                                   block and its length: ");
    scanf("%d%d", &st, &len);
    k = len;
    if (pages[st] == 0)
    {
        for (j = st; j < (st + k); j++)
        {
            if (pages[j] == 0)
            {
                pages[j] = 1;
                printf("%d------>%d\n", j, pages[j]);
            }
            else
            {
                printf("The block %d is
                                  already allocated \n", j);
                k++;
            }
        }
    }
    else
        printf("The block %d is already allocated \n", st);
    printf("Do you want to enter more files? \n");
    printf("Enter 1 for Yes, Enter 0 for No: ");
    scanf("%d", &c);
    if (c==1) recursivePart(pages);
    else exit(0);
    return;
}
```



```c
int main()
{
    int pages[50], p, a;

    for (int i = 0; i < 50; i++)
        pages[i] = 0;
    printf("Enter the number of blocks already allocated: ")
;
    scanf("%d", &p);
    printf("Enter the blocks already allocated: ");
    for (int i = 0; i < p; i++)
    {
        scanf("%d", &a);
        pages[a] = 1;
    }

    recursivePart(pages);
    return 0;
}
```

Output:-

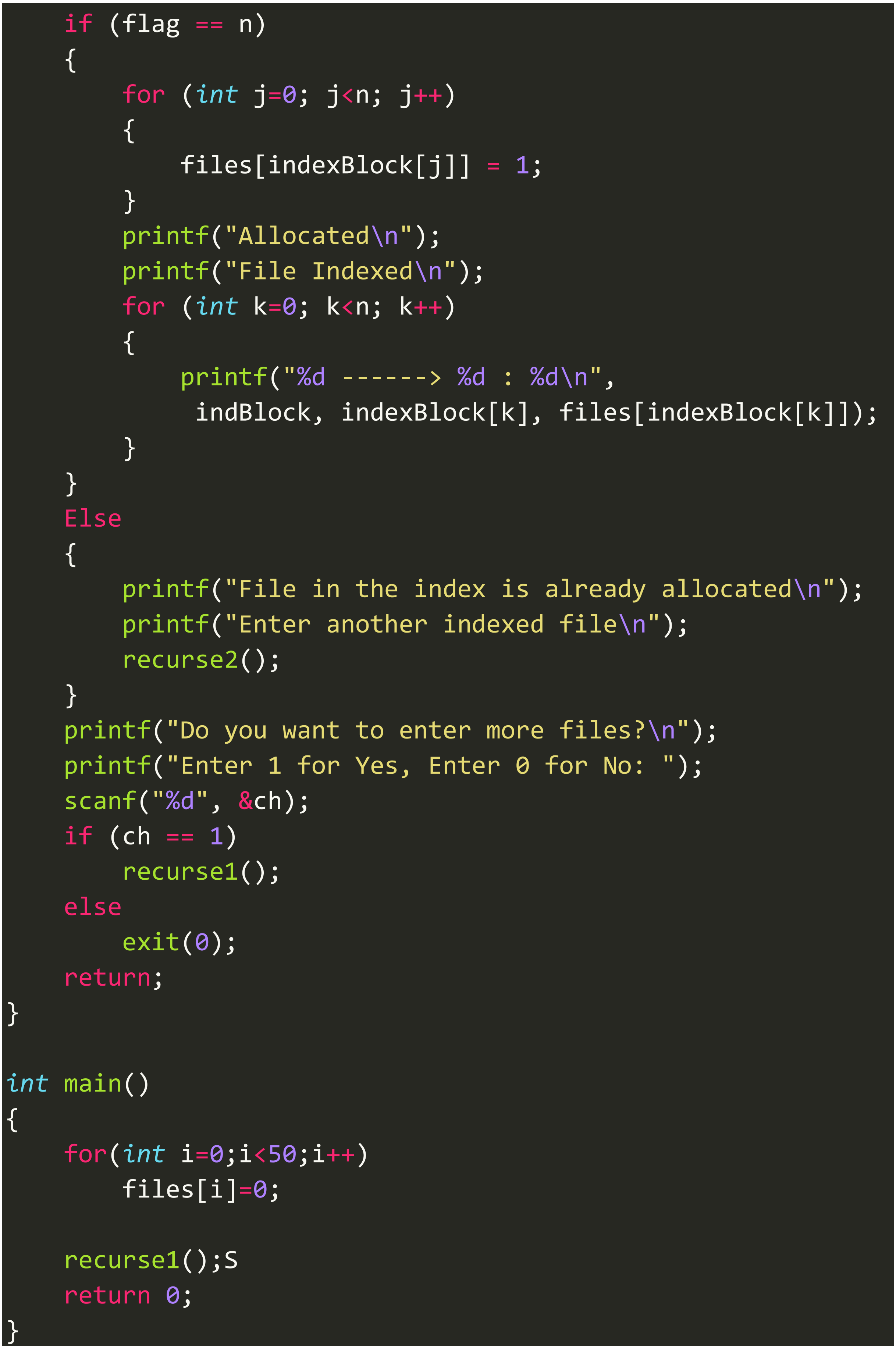


```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ gcc Linked.c -o Linked.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ ./Linked.exe
Enter the number of blocks already allocated: 3
Enter the blocks already allocated: 1 4 6
Enter the index of the starting block and its length: 10 3
10------>1
11------>1
12------>1
Do you want to enter more files?
Enter 1 for Yes, Enter 0 for No: 1
Enter the index of the starting block and its length: 3 2
3------>1
The block 4 is already allocated
5------>1
Do you want to enter more files?
Enter 1 for Yes, Enter 0 for No: 0
```

c. Indexed

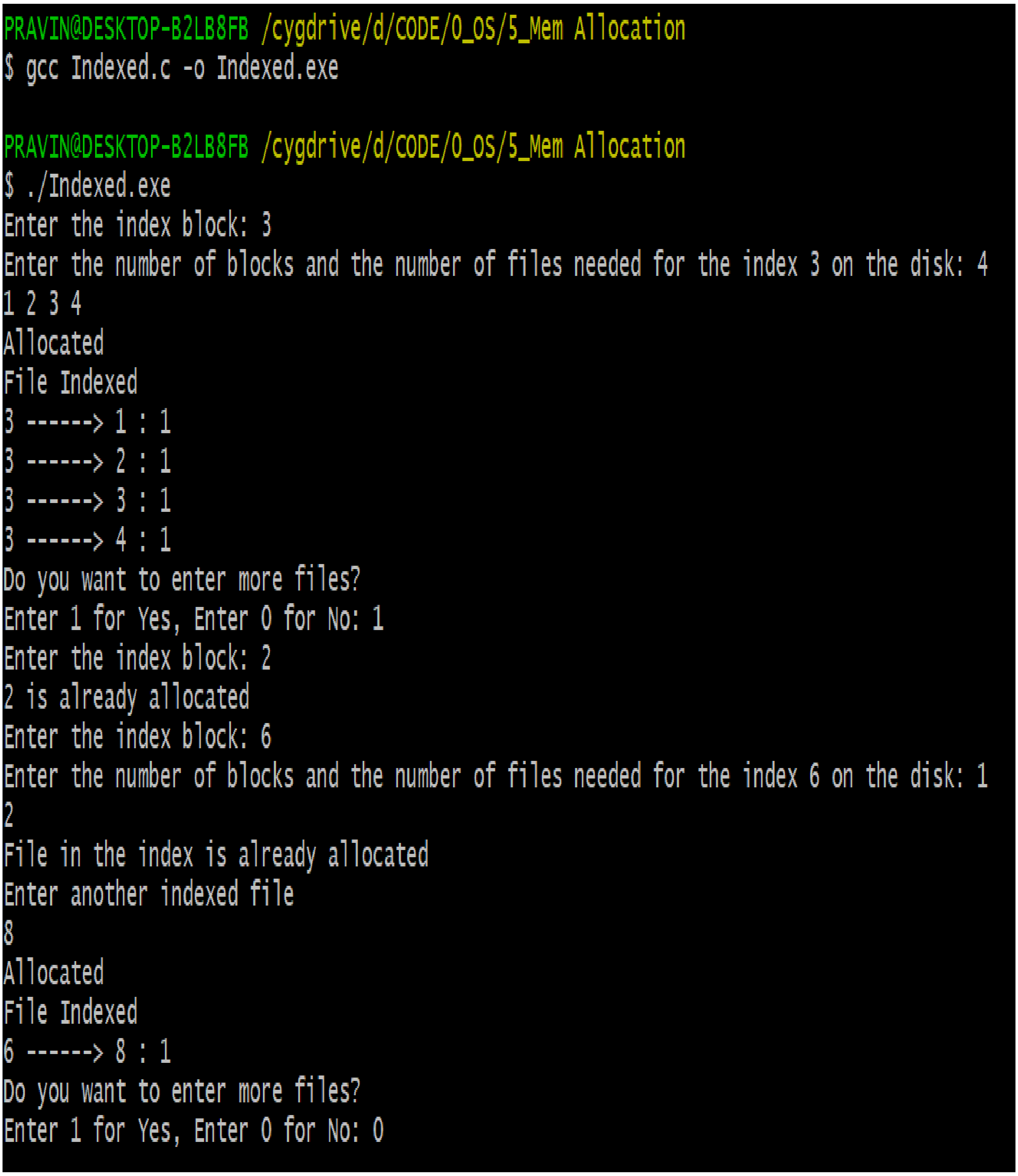
Code:-

```
#include <stdio.h>
#include <stdlib.h>
int files[50], indexBlock[50], indBlock, n;
void recurse1();
void recurse2();

void recurse1()
{
    printf("Enter the index block: ");
    scanf("%d", &indBlock);
    if (files[indBlock] != 1)
    {
        printf("Enter the number of blocks and the
    number of files needed for the index %d on the disk: ",
                                                       indBlock);
        scanf("%d", &n);
    }
    Else
    {
        printf("%d is already allocated\n", 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("Allocated\n");
        printf("File Indexed\n");
        for (int k=0; k<n; k++)
        {
            printf("%d ------> %d : %d\n",
             indBlock, indexBlock[k], files[indexBlock[k]]);
        }
    }
    Else
    {
        printf("File in the index is already allocated\n");
        printf("Enter another indexed file\n");
        recurse2();
    }
    printf("Do you want to enter more files?\n");
    printf("Enter 1 for Yes, Enter 0 for No: ");
    scanf("%d", &ch);
    if (ch == 1)
        recurse1();
    else
        exit(0);
    return;
}

int main()
{
    for(int i=0;i<50;i++)
        files[i]=0;

    recurse1();S
    return 0;
}
```

Output:-



```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ gcc Indexed.c -o Indexed.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/CODE/0_OS/5_Mem Allocation
$ ./Indexed.exe
Enter the index block: 3
Enter the number of blocks and the number of files needed for the index 3 on the disk: 4
1 2 3 4
Allocated
File Indexed
3 ------> 1 : 1
3 ------> 2 : 1
3 ------> 3 : 1
3 ------> 4 : 1
Do you want to enter more files?
Enter 1 for Yes, Enter 0 for No: 1
Enter the index block: 2
2 is already allocated
Enter the index block: 6
Enter the number of blocks and the number of files needed for the index 6 on the disk: 1
2
File in the index is already allocated
Enter another indexed file
8
Allocated
File Indexed
6 ------> 8 : 1
Do you want to enter more files?
Enter 1 for Yes, Enter 0 for No: 0
```