ITE2002-OPERATING SYSTEM

LAB

WINTER SEM 20-21

FAT

Program 8

Name : Pravin G

Reg No :19BIT0393

Slot:L41+42

Qno1

Algorithm:

Step 1-read:
Read number of partitions blocks
Read sizes of each partition blocks
Read number of process
Read sizes of each process

Step 2: best fit
For each process find a block which is just greater than process size.

Then allocate process in that block

new space of partition = old space of partition – process size

if not such block available that process have to wait

do it for all process

step 3 :worst fit
For each process find a block which is largest among available and that is greater than process size.

Then allocate process in that block

new space of partition = old space of partition – process size

if not such block available that process have to wait

do it for all process

Code:

```
#include<stdio.h>
int nm,np,i,j;
int M[10],P[10],FF[10],BF[10],WF[10];

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+1,P[i]);
    else {
      printf("\nProcess P%d-(%dK) is Put in Memory M%d-
(%dK)",i+1,P[i],idx+1,BF[idx]);
      printf("(New Partition %dK-
%dK = %dK)",BF[idx],P[i],BF[idx]-P[i]);
      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])
        idx=j;
```

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

  bestFit();
  worstFit();
}
```

Excution

```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/OS/19BIT0393/fat
$ gcc fat1.c -o fat1.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/OS/19BIT0393/fat
$ ./fat1.exe
```

Input:

```
Enter no of Memory Partitions : 5
Enter Memory size of Each Partition : 300 600 400 500 600
Enter no of Processes : 4
Enter Memory size of Each Process : 320 430 110 520
```

Output:

Best fit

```
Best Fit :
Process P1-(320K) is Put in Memory M3-(400K)(New Partition 400K-320K = 80K)
Process P2-(430K) is Put in Memory M4-(500K)(New Partition 500K-430K = 70K)
Process P3-(110K) is Put in Memory M1-(300K)(New Partition 300K-110K = 190K)
Process P4-(520K) is Put in Memory M2-(600K)(New Partition 600K-520K = 80K)
```

Worst fit

```
Worst Fit :
Process P1-(320K) is Put in Memory M2-(600K)(New Partition 600K-320K = 280K)
Process P2-(430K) is Put in Memory M5-(600K)(New Partition 600K-430K = 170K)
Process P3-(110K) is Put in Memory M4-(500K)(New Partition 500K-110K = 390K)
Process P4-(520K) must wait
```

Qno2

Algorithm:

Step 1: read

Read number of refernce string elements

Read reference string charter by space separated

Read no of frames

Step 2 : Optimal algo

Take a reference from refernce string

If it is available in frame print hit and increment count of hit

Otherwise

Select a frame which is not going to used for long time

Replace refernce to that frame and print as page fault

Do it for all references
Step 3:print
Print no of hits,no of faults, hit ratio

Code:

```
#include<stdio.h>
char RS[50];

int nr,nf,i,j;

char F[5]={'_','_','_','_','_'};
int hit=0;
int isrt=0;
int flag1=0,flag2=0;
int k,max;

void optimal()
{
  printf("\n\n\nOptimal :- \n");
  printf("Pages   ");
  for(i=0;i<nf;i++)
    printf("F%d ",i+1);
  printf("-Hit/Page Fault-\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])
      {
        for(j=0;j<nf;j++)
          printf("%c  ",F[j]);
        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("-- Page Fault --\n");
  }
  printf("\nNo of Page Faults = %d",nr-hit);
  printf("\nNo of Hits = %d",hit);
  printf("\nHit Ratio = %.2f",hit/(float)nr);
}
```

```c
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);


  optimal();

}
```

Exection

```
PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/OS/19BIT0393/fat
$ gcc fat2.c -o fat2.exe

PRAVIN@DESKTOP-B2LB8FB /cygdrive/d/OS/19BIT0393/fat
$ ./fat2.exe
```

Input:

```
Enter number of References : 21
Enter Reference String : 6 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 6 0 1 6
Enter no of Frames : 3
```

Output:

```
Optimal :-
Pages   F1 F2 F3 -Hit/Page Fault-
6       6  _  _  -- Page Fault --
0       6  0  _  -- Page Fault --
1       6  0  1  -- Page Fault --
2       2  0  1  -- Page Fault --
0       2  0  1  ------ Hit -----
3       2  0  3  -- Page Fault --
0       2  0  3  ------ Hit -----
4       2  4  3  -- Page Fault --
2       2  4  3  ------ Hit -----
3       2  4  3  ------ Hit -----
0       2  0  3  -- Page Fault --
3       2  0  3  ------ Hit -----
2       2  0  3  ------ Hit -----
1       2  0  1  -- Page Fault --
2       2  0  1  ------ Hit -----
0       2  0  1  ------ Hit -----
1       2  0  1  ------ Hit -----
6       6  0  1  -- Page Fault --
0       6  0  1  ------ Hit -----
1       6  0  1  ------ Hit -----
6       6  0  1  ------ Hit -----

No of Page Faults = 9
No of Hits = 12
Hit Ratio = 0.57
```