Exp. No:10	MEMORY MANAGEMENT SCHEME –FIRST FIT, BEST FIT, WORST FIT

Aim:

To write a C program to implement the Memory Management concept using the first fit, Best fit and Worst fit.

Algorithm:

Step1: Start the program.

Step2: Check the conditions in the program and run the program.

Step3: Enter the number of blocks and number of processors.

Step4: Enter the Size of each processor and size of each block for the given input.

Step5: Print the values of First fit, Best fit and Worst fit for the given input.

Step6: Stop the program.

Program:

```
#include<stdio.h>
#include<string.h>
void firstFit(int bsf[], int m,int psf[], int n)
{
int alloc[n];
for(int i=0;i<n;i++)
alloc[i]=-1;
for (int i = 0; i < n; i++)
{
for (int j = 0; j < m; j++)
{
if (bsf[j] >= psf[i])
{
alloc[i] = j;
bsf[j] -= psf[i];
break;
}
}
}
```

```
printf("\nProcess No.\tProcess Size\tBlock no.\n");
for (int i = 0; i < n; i++)
{
printf("%d\t\t%d\t\t",i+1,psf[i]);
if (alloc[i] != -1)
printf("%d",alloc[i] + 1);
else
printf("Not Allocated");
printf("\n");
}
}
void bestFit(int bsb[], int m,int psb[], int n)
{
int alloc[n];
for (int i = 0; i < n; i++)
alloc[i] = -1;
for (int i = 0; i < n; i++)
{
int bestIdx = -1;
for (int j = 0; j < m; j++)
{
if (bsb[j] >= psb[i])
{
if (bestIdx == -1)
bestIdx = j;
else if (bsb[bestIdx]> bsb[j])
bestIdx = j;
}
}
if (bestIdx != -1)
```

```
{
alloc[i] = bestIdx;
bsb[bestIdx] -=psb[i];
}
}
printf("\nProcess No.\tProcess\tSize\tBlock no.\n");
for (int i = 0; i < n; i++)
{
printf("%d\t\t%d\t\t",i+1, psb[i]);
if (alloc[i] != -1)
printf("%d",alloc[i] + 1);
else
printf("Not Allocated");
printf("\n");
}
}
void worstFit(int bsw[], int m,int psw[], int n)
{
int alloc[n];
for (int i = 0; i < n; i++)
alloc[i] = -1;
for (int i = 0; i < n; i++)
{
int wstIdx = -1;
for (int j = 0; j < m; j++)
{
if (bsw[j] >= psw[i])
{
if (wstIdx == -1)
wstIdx = j;
```

```
else if (bsw[wstIdx] < bsw[j])
wstIdx = j;
}
}
if (wstIdx != -1)
{
alloc[i] = wstIdx;
bsw[wstIdx] -= psw[i];
}
}
printf("\nProcess No\tProcess Size\tBlock no\n");
for (int i = 0; i < n; i++)
{
printf("%d\t\t%d\t\t",i+1,psw[i]);
if (alloc[i] != -1)
printf("%d",alloc[i] + 1);
else
printf("Not Allocated");
printf("\n");
}
}
void main()
{
int m,n,bsfirst[10],bsworst[10],bsbest[10],psfirst[10],psworst[10],psbest[10];
printf("Enter no.of blocks : ");
scanf("%d",&m);
printf("Enter no.of processors: ");
scanf("%d",&n);
printf("Enter the size of Each Block\n----------\n");
for(int i=0;i<m;i++)
```

```
{
printf("Block [%d] : ",i);
scanf("%d",&bsfirst[i]);
bsworst[i]=bsfirst[i];
bsbest[i]=bsfirst[i];
}
printf("Enter the size of Each Processor\n----------\n");
for(int i=0;i<n;i++)
{
printf("Processor [%d] : ",i);
scanf("%d",&psfirst[i]);
psworst[i]=psfirst[i];
psbest[i]=psfirst[i];
}
printf("\n-------FIRST FIT-------\n");
firstFit(bsfirst, m, psfirst, n);
printf("\n-------BEST FIT-------\n");
bestFit(bsworst, m, psworst, n);
printf("\n-------WORST FIT-------\n");
worstFit(bsbest, m, psbest, n);
}
```

Output:

```
hari@HARIPRIYA:~$ ./a.out
Enter no.of blocks : 3
Enter no.of processors: 3
Enter the size of Each Block
----------
Block [0] : 1
Block [1] : 2
Block [2] : 3
Enter the size of Each Processor
----------
Processor [0] : 5
Processor [1] : 6
Processor [2] : 7

-------FIRST FIT-------

Process No.     Process Size    Block no.
1               5               Not Allocated
2               6               Not Allocated
3               7               Not Allocated

-------BEST FIT-------

Process No.     Process Size    Block no.
1               5               Not Allocated
2               6               Not Allocated
3               7               Not Allocated

-------WORST FIT-------

Process No      Process Size    Block no
1               5               Not Allocated
2               6               Not Allocated
3               7               Not Allocated
```

Exp. No: 11	FILE ALLOCATION STRATEGIES – SEQUENTIAL ALLOCATION

Aim:

To implement file allocation technique using sequential allocation.

Algorithm:

Step1: Start the program.

Step2: Check the conditions of the program and run the program.

Step3: Enter the file name and the starting block of the file.

Step4: Enter the length of the file and if the block is empty and the file is allocated in the Disk.

Step5:print the blocks allocated to the file.

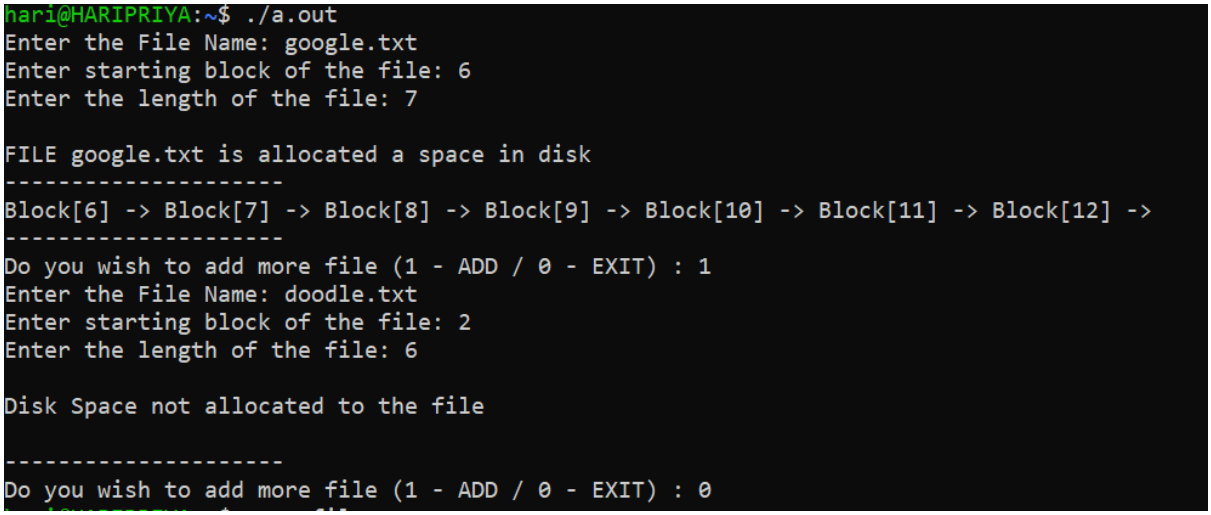
Step6: stop the program.

Program:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
char fn[20];
for(i=0;i<50;i++)
f[i]=0;
do
{
count=0;
printf("Enter the File Name: ");
scanf("%s",fn);
printf("Enter starting block of the file: ");
scanf("%d", &st);
printf("Enter the length of the file: ");
scanf("%d",&len);
for(k=st;k<(st+len);k++)
```

```
if(f[k]==0)
count++;
if(len==count)
{
if(j!=(st+len-1))
printf("\nFILE %s is allocated a space in disk\n----------\n",fn);
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("Block[%d] -> ",j);
}
}
else
printf("\nDisk Space not allocated to the file\n");
printf("\n----------\nDo you wish to add more file (1 - ADD / 0 - EXIT) : ");
scanf("%d", &c);
}while(c==1);
}
```

Output:



Exp. No: 12	IMPLEMENTATION OF FIFO PAGE REPLACEMENT TECHNIQUE

Aim:

To implement the FIFO page replacement technique.

Algorithm:

Step1: Start the program.

Step2: Check the conditions and run the program.

Step3: Enter the number of pages needed and number of page number .

Step4: Enter the number of frames needed for the program.

Step6: Print the allocated page number and page frames.

Step7: Stop the program.

Program:

```
#include<stdio.h>
int main(){
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\nENTER THE NUMBER OF PAGES --> ");
scanf("%d",&n);
printf("\nENTER THE PAGE NUMBER --> ");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("\nENTER THE NUMBER OF FRAMES -->");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]= -1;
j=0;
printf("Page Number\tPage frames\n---------------------------------------\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t",a[i]);
avail = 0;
```

```
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if (avail==0)
{
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);
}
printf("\n");
}
printf("Page Fault = %d\n",count);
return 0;
}
```

Output:

```
hari@HARIPRIYA:~$ ./a.out

ENTER THE NUMBER OF PAGES --> 5

ENTER THE PAGE NUMBER --> 4
5
6
7
8

ENTER THE NUMBER OF FRAMES -->2
Page Number     Page frames
---------------------------------------
4               4       -1
5               4       5
6               6       5
7               6       7
8               8       7
Page Fault = 5
```