OS LAB-1BM21CS203

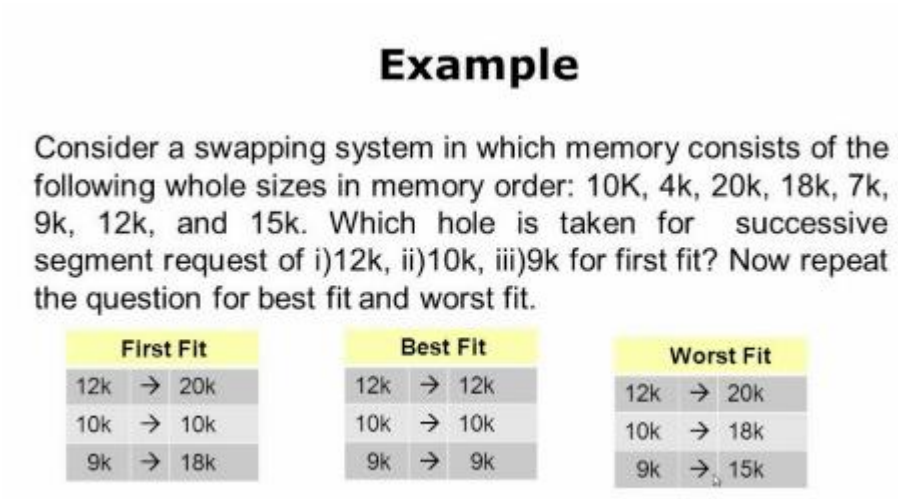
Write a C program to simulate the following contiguous memory allocation techniques

a) Worst-fit

b) Best-fit

c) First-fit

Simulate the following situation:



Example

Consider a swapping system in which memory consists of the following whole sizes in memory order: 10K, 4k, 20k, 18k, 7k, 9k, 12k, and 15k. Which hole is taken for successive segment request of i)12k, ii)10k, iii)9k for first fit? Now repeat the question for best fit and worst fit.

First Fit		
12k	→	20k
10k	→	10k
9k	→	18k

Best Fit		
12k	→	12k
10k	→	10k
9k	→	9k

Worst Fit		
12k	→	20k
10k	→	18k
9k	→	15k

CODE:

```
#include<stdio.h>
#include<conio.h>
#define max 25
void firstfit()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];

printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
```

```
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
{
ff[i]=j;
break;
}
}
}
frag[i]=temp;
bf[ff[i]]=1;
}
printf("\nFile_size:\tBlock_size:");
for(i=1;i<=nf;i++)
```

```
printf("\n%d\t\t%d",f[i],b[ff[i]]);
}

void bestfit()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];

printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
```

```
temp=b[j]-f[i];
if(temp>=0)
if(lowest>temp)
{
ff[i]=j;

lowest=temp;
}
}
}
frag[i]=lowest;
bf[ff[i]]=1;
lowest=10000;
}
printf("\nFile Size:\tBlock Size:");
for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d",f[i],b[ff[i]]);
}

void worstfit()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];

printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
```

```
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{

for(j=1;j<=nb;j++)
{
if(bf[j]!=1) //if bf[j] is not allocated
{
temp=b[j]-f[i];
if(temp>=0)
if(highest<temp)
{
ff[i]=j;
highest=temp;
}
}
}
frag[i]=highest;
bf[ff[i]]=1;
highest=0;
}
```

```
printf("\nFile_size:\tBlock_size:");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d",f[i],b[ff[i]]);
}

void main()
{
int c;
while(1)
{
printf("\n1.First Fit 2.Best Fit 3.Worst Fit 4.Exit");
printf("\nEnter choice:");
scanf("%d",&c);
switch(c)
{
case 1:firstfit();
break;
case 2:bestfit();
break;
case 3:worstfit();
break;
case 4:exit(0);
default:printf("Invalid choice");
}
}
}
```

OUTPUT:

"C:\Users\STUDENT\Desktop\os lab 1bm21cs203\memory.exe"

```
1.First Fit 2.Best Fit 3.Worst Fit 4.Exit
Enter choice:1

Enter the number of blocks:8
Enter the number of files:3

Enter the size of the blocks:-
Block 1:10000
Block 2:4000
Block 3:20000
Block 4:18000
Block 5:7000
Block 6:9000
Block 7:12000
Block 8:15000
Enter the size of the files :-
File 1:12000
File 2:10000
File 3:9000

File_size:      Block_size:
12000           20000
10000           10000
9000            18000
1.First Fit 2.Best Fit 3.Worst Fit 4.Exit
Enter choice:2

Enter the number of blocks:8
Enter the number of files:3

Enter the size of the blocks:-
Block 1:10000
Block 2:4000
Block 3:20000
Block 4:18000
Block 5:7000
Block 6:9000
Block 7:12000
Block 8:15000
Enter the size of the files :-
File 1:12000
File 2:10000
File 3:9000

File Size:      Block Size:
12000           12000
10000           10000
9000            9000
```

"C:\Users\STUDENT\Desktop\os lab 1bm21cs203\memory.exe"

```
12000           12000
10000           10000
9000            9000
1.First Fit 2.Best Fit 3.Worst Fit 4.Exit
Enter choice:3

Enter the number of blocks:8
Enter the number of files:3

Enter the size of the blocks:-
Block 1:10000
Block 2:4000
Block 3:20000
Block 4:18000
Block 5:7000
Block 6:9000
Block 7:12000
Block 8:15000
Enter the size of the files :-
File 1:12000
File 2:10000
File 3:9000

File_size:      Block_size:
12000           20000
10000           18000
9000            15000
1.First Fit 2.Best Fit 3.Worst Fit 4.Exit
Enter choice:4

Process returned 0 (0x0)   execution time : 274.045 s
Press any key to continue.
```