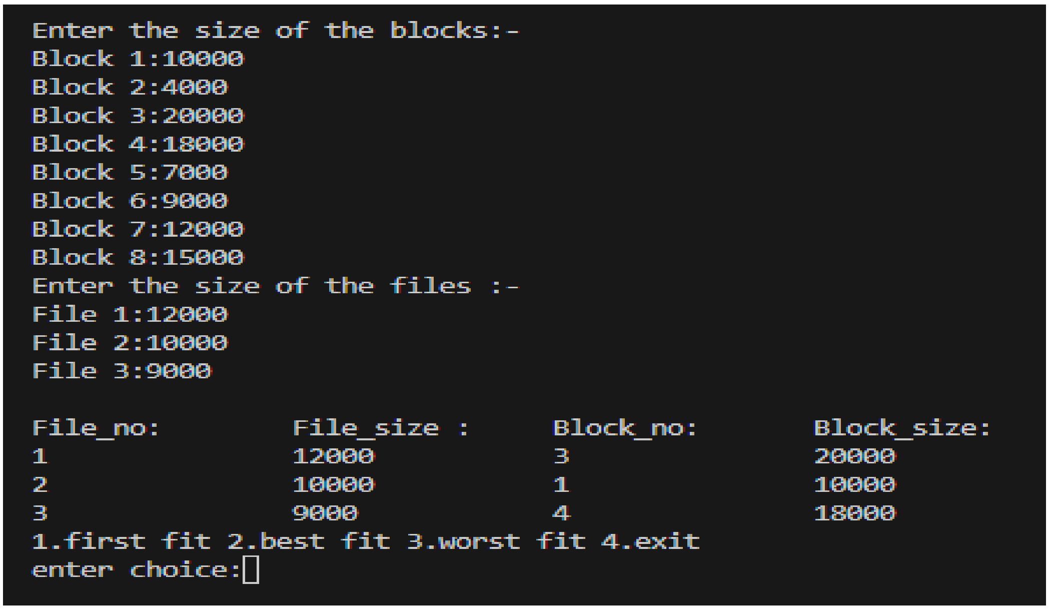
MEMORY MANAGEMENT
BEST FIT, WORST FIT, FIRST FIT

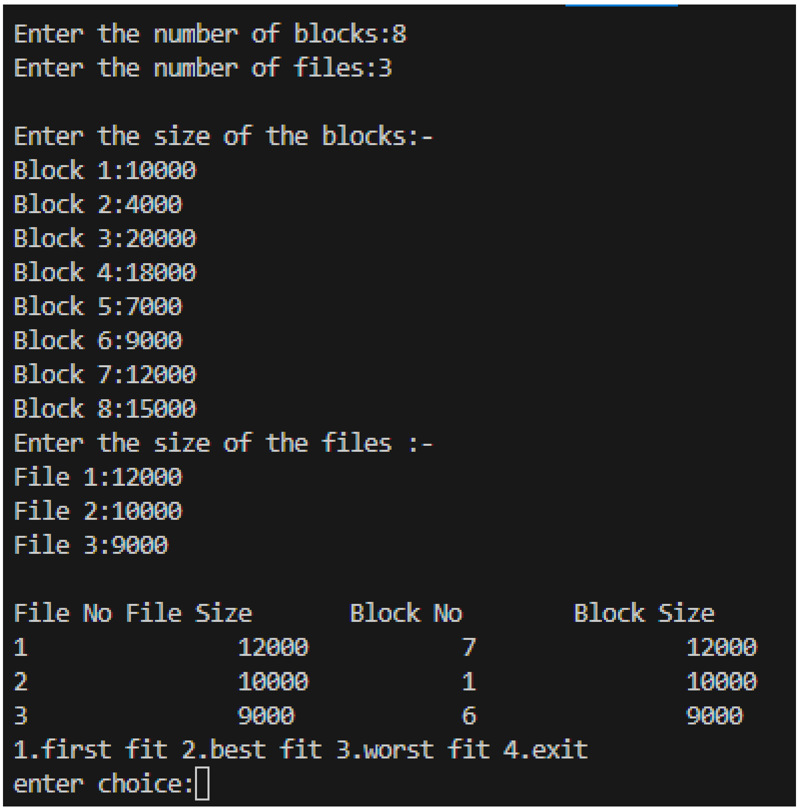
```
#include<stdio.h>
#include<conio.h>
#define max 25
void firstfit()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];

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_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[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 No\tFile Size \tBlock No\tBlock Size\tFragment");
for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}

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

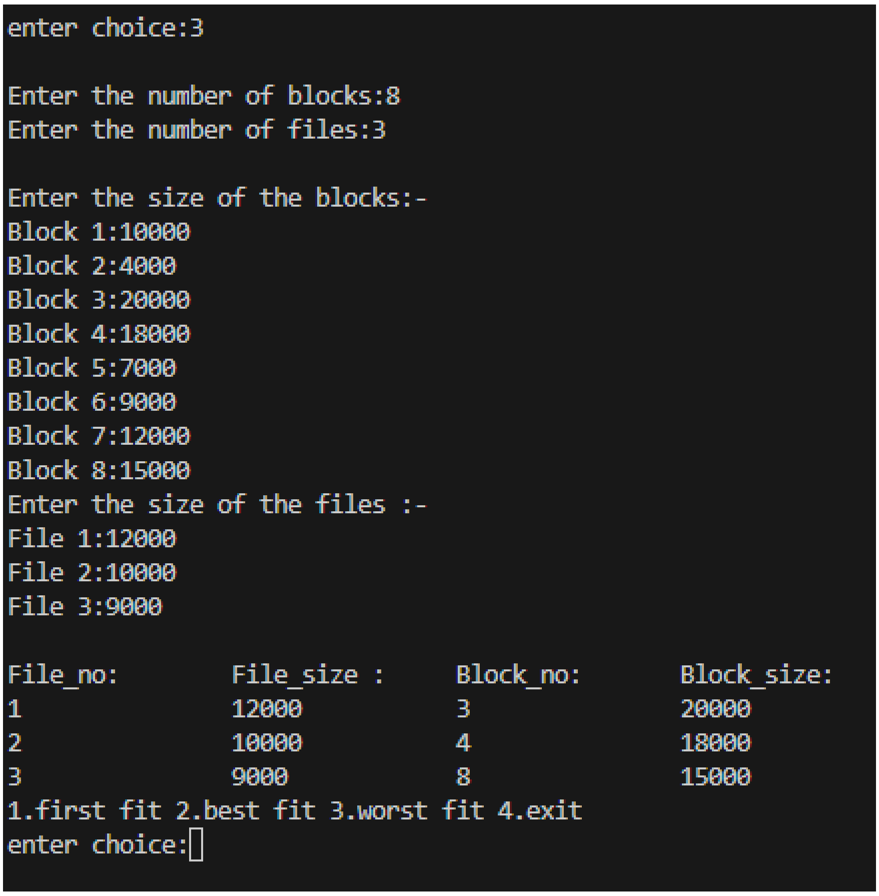
```
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %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_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[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");
}
}
}
```



```
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_no:        File_size :     Block_no:       Block_size:
1               12000           3               20000
2               10000           1               10000
3               9000            4               18000
1.first fit 2.best fit 3.worst fit 4.exit
enter choice:
```



```
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 No File Size       Block No        Block Size
1               12000           7               12000
2               10000           1               10000
3               9000            6               9000
1.first fit 2.best fit 3.worst fit 4.exit
enter choice:
```



```
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_no:        File_size :     Block_no:       Block_size:
1               12000           3               20000
2               10000           4               18000
3               9000            8               15000
1.first fit 2.best fit 3.worst fit 4.exit
enter choice:
```