EX NO: 8B	**MEMORY ALLOCATION METHODS FOR FIXED PARTITION**
DATE:	

I. FIRST FIT ALLOCATION

AIM:

To allocate memory requirements for processes using first fit allocation.

ALGORITHM:

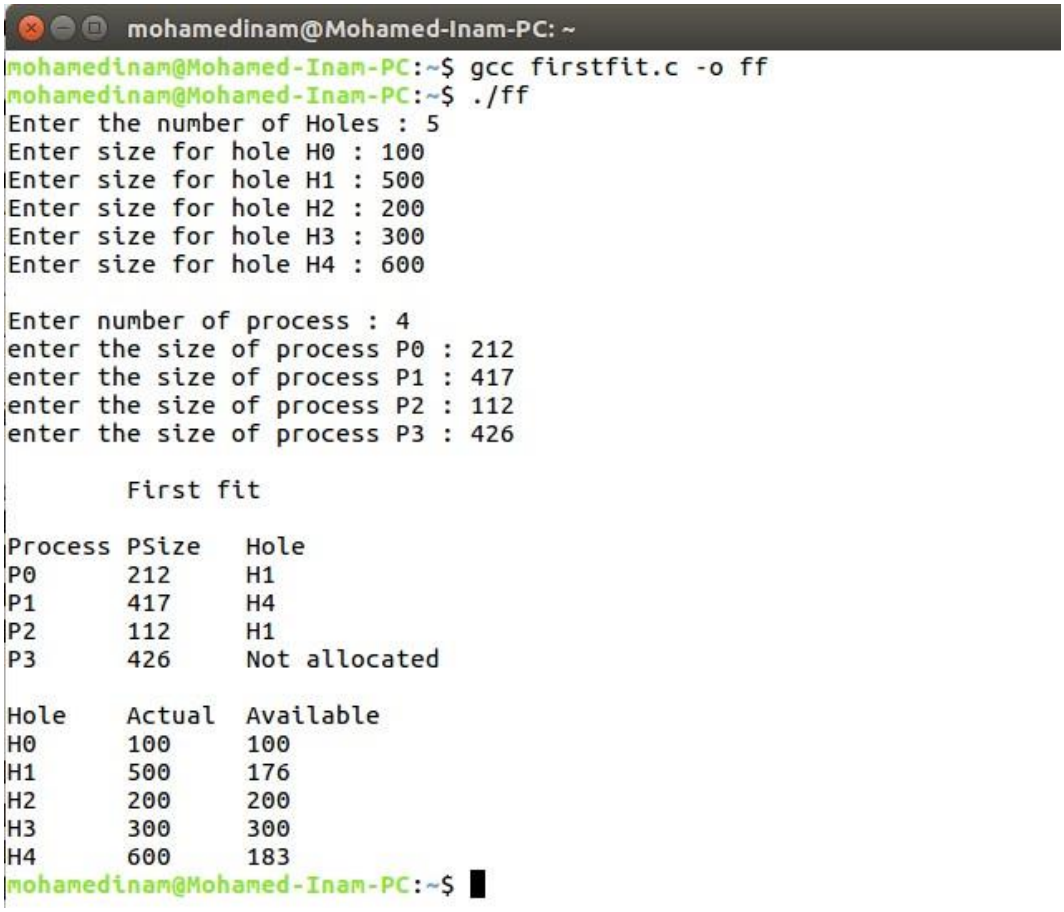
1. Declare structures *hole* and *process* to hold information about set of holes and processes respectively.
2. Get number of holes, say *nh*.
3. Get the size of each hole
4. Get number of processes, say *np*.
5. Get the memory requirements for each process.
6. Allocate processes to holes, by examining each hole as follows:
   a. If hole size>process size then
   i. Mark process as allocated to that hole. ii.
   Decrement hole size by process size.
   b. Otherwise check the next from the set of holes
7. Print the list of process and their allocated holes or unallocated status.
8. Print the list of holes, their actual and current availability.
9. Stop the program.

80

PROGRAM

```
#include<stdio.h>
struct process {
int size; int flag; int holeid;
}p[10]; struct hole { int size; int
actual; }h[10]; main() { int i,np,nh,j;
printf("EnterthenumberofHoles:");
scanf("%d",&nh);
for(i=0;i<nh;i++){
printf("EntersizeforholeH%d:",i);
scanf("%d",&h[i].size);
h[i].actual=h[i].size;
}
printf("\nEnternumberofprocess:");
scanf("%d",&np); for(i=0;i<np;i++){
printf("enterthesizeofprocessP%d:",i);
scanf("%d",&p[i].size); p[i].flag=0;
} for(i=0;i<np;i++){
for(j=0;j<nh;j++){
if(p[i].flag!=1){
if(p[i].size<=h[j].size){
p[i].flag=1; p[i].holeid=j; h[j].size-=p[i].size;
}}}}

printf("\n\tFirstfit\n");
printf("\nProcess\tPSize\tHole");
for(i=0;i<np;i++){ if(p[i].flag!=1)
printf("\nP%d\t%d\tNotallocated",i,p[i].size); else
printf("\nP%d\t%d\tH%d",i,p[i].size,p[i].holeid);
}
printf("\n\nHole\tActual\tAvailable"); for(i=0;i<nh;i++)
printf("\nH%d\t%d\t%d",i,h[i].actual,h[i].size);
printf("\n");
}
```

OUTPUT:


mohamedinam@Mohamed-Inam-PC: ~
mohamedinam@Mohamed-Inam-PC:~$ gcc firstfit.c -o ff
mohamedinam@Mohamed-Inam-PC:~$ ./ff
Enter the number of Holes : 5
Enter size for hole H0 : 100
Enter size for hole H1 : 500
Enter size for hole H2 : 200
Enter size for hole H3 : 300
Enter size for hole H4 : 600
Enter number of process : 4
enter the size of process P0 : 212
enter the size of process P1 : 417
enter the size of process P2 : 112
enter the size of process P3 : 426
First fit
Process PSize Hole
P0 212 H1
P1 417 H4
P2 112 H1
P3 426 Not allocated
Hole Actual Available
H0 100 100
H1 500 176
H2 200 200
H3 300 300
H4 600 183
mohamedinam@Mohamed-Inam-PC:~$

RESULT:

Thus processes were allocated memory using first fit method.

II. WORST FIT ALLOCATION

AIM:

To allocate memory requirements for processes using worst fit allocation.

ALGORITHM:

Step 1: Start the program.
Step 2: Input memory blocks and processes with sizes.
Step 3: Initialize all memory blocks as free. Step 4: Start by picking each process and find the maximum block size that can be assigned to current process i.e., find max(bockSize[1], blockSize[2],.....blockSize[n]) > processSize[current], if found then assign it to the current process.
Step 5: If not then leave that process and keep checking the further processes. Step 6: Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h> #define
max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;  static
int bf[max],ff[max];
clrscr();
printf("\n\tMemory Management Scheme - First Fit");
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]);  getch();
}
```

INPUT

Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:- Block
1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:- File
1: 1
File 2: 4

OUTPUT

File No	File Size	Block No	Block Size	Fragment
1	1	1	5	4
2	4	3	7	3

RESULT

Thus processes were allocated memory using worst fit method.

III. BEST FIT ALLOCATION

AIM:

To allocate memory requirements for processes using best fit allocation.

ALGORITHM:

1. Declare structures hole *and* process *to* hold information about set of holes and processes respectively.
2. Get number of holes, say *nh*.
3. Get the size of each hole
4. Get number of processes, say *np*.
5. Get the memory requirements for each process.
6. Allocate processes to holes, by examining each hole as follows:
   a. Sort the holes according to their sizes in ascending order
   b. If hole size>process size then
      i. Mark process as allocated to that hole.
      ii. Decrement hole size by process size.

c. Otherwise check the next from the set of sorted hole.
7. Print the list of process and their allocated holes or unallocated status.
8. Print the list of holes, their actual and current availability.
9. Stop

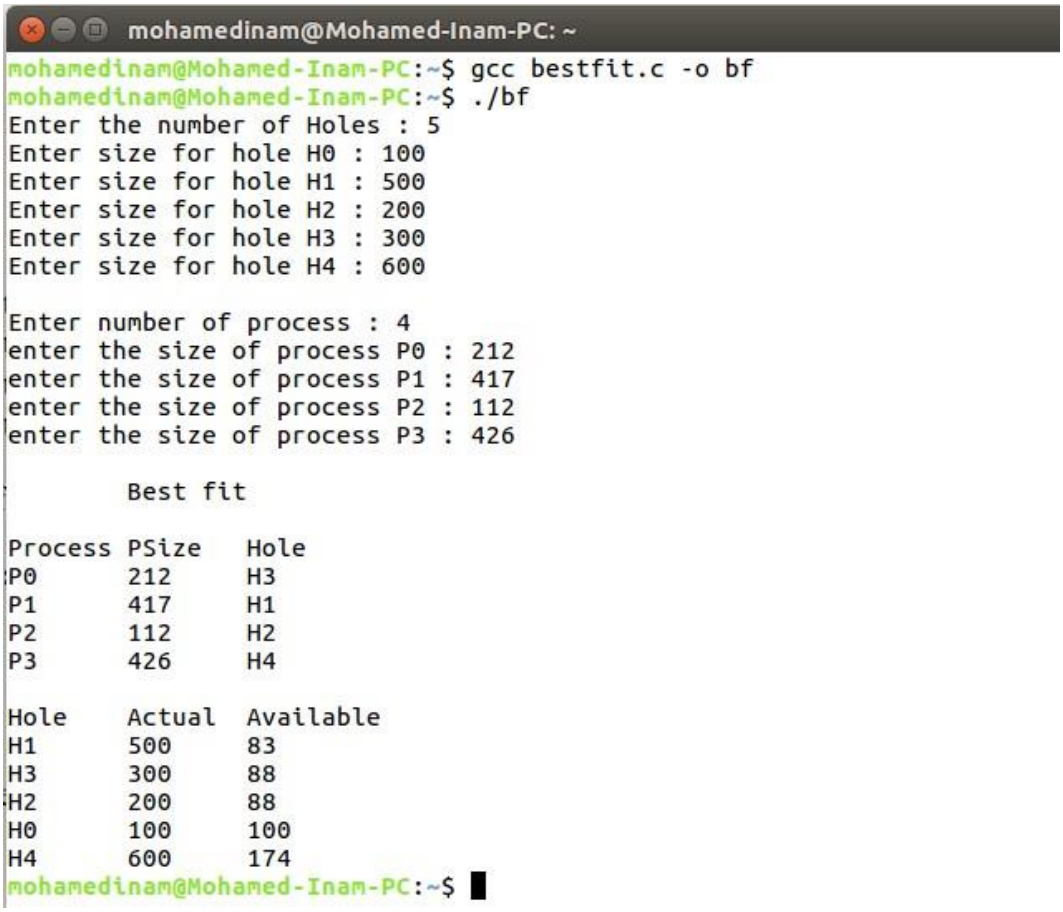
PROGRAM:

```
#include<stdio.h>

struct process
{ int size;  int
flag;  int
holeid; }p[10];

struct hole
{ int hid;
int size; int
actual;
}h[10];

main() { int i,np,nh,j; void
bsort(structhole[],int); printf("Enter
```

```
the number of Holes:");
scanf("%d",&nh); for(i=0;i<nh;i++) {
printf("Enter size for holeH%d:",i);
scanf("%d",&h[i].size);
h[i].actual=h[i].size;  h[i].hid=i;
}
printf("\nEnter number of process:");
scanf("%d",&np); for(i=0;i<np;i++) {
printf("enter the size of processP%d:",i);
scanf("%d",&p[i].size); p[i].flag=0;
}

for(i=0;i<np;i++)
{ bsort(h,nh);
for(j=0;j<nh;j++)
{ if(p[i].flag!=1) {
if(p[i].size<=h[j].size)
{ p[i].flag=1;
p[i].holeid=h[j].hid;
h[j].size-=p[i].size;
}
}
}
}

printf("\n\tBestfit\n");
printf("\nProcess\tPSize\tHole"); for(i=0;i<np;i++)
{ if(p[i].flag!=1)
printf("\nP%d\t%d\tNotallocated",i,p[i].size);
else
printf("\nP%d\t%d\tH%d",i,p[i].size,p[i].holeid);
}

printf("\n\nHole\tActual\tAvailable"); for(i=0;i<nh;i++)
printf("\nH%d\t%d\t%d",h[i].hid,h[i].actual, h[i].size);
printf("\n");
}

Void bsort(structholebh[],intn)
{ struct
holetemp;
int i,j;
for(i=0;i<n-1;i++)
```

```
{
for(j=i+1;j<n;j++)
{
if(bh[i].size>bh[j].size)
{ temp=bh[i];
bh[i]=bh[j];
bh[j]=temp;
}
}
}
}
```

OUTPUT:


mohamedinam@Mohamed-Inam-PC: ~
mohamedinam@Mohamed-Inam-PC:~$ gcc bestfit.c -o bf
mohamedinam@Mohamed-Inam-PC:~$ ./bf
Enter the number of Holes : 5
Enter size for hole H0 : 100
Enter size for hole H1 : 500
Enter size for hole H2 : 200
Enter size for hole H3 : 300
Enter size for hole H4 : 600
Enter number of process : 4
enter the size of process P0 : 212
enter the size of process P1 : 417
enter the size of process P2 : 112
enter the size of process P3 : 426
Best fit
Process PSize Hole
P0 212 H3
P1 417 H1
P2 112 H2
P3 426 H4
Hole Actual Available
H1 500 83
H3 300 88
H2 200 88
H0 100 100
H4 600 174
mohamedinam@Mohamed-Inam-PC:~$

RESULT:

Thus processes were allocated memory using best fit method.