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

a) Worst fit b) Best fit c) first fit

a)
```
#include <stdio.h>
void worstfit (int blocksize[], int blocks, int process
                                        size[],
int process)

{
int allocation [processes];
int occupied [blocks];
for (int i = 0; i < process ; i++) {
  allocation [i] = -1;
}
for (i = 0; i < blocks ; i++) {
  occupied [i] = 0;
}
for (i = 0; i < process ; i++){
int indexplaced = -1;
for (j = 0; j < blocks ; j++){
if [blocksize[j] >= processsize[i] && !occupied [j])
{
  if (indexplaced != -1)
    indexplaced = j;
  else if (blocksize[indexplaced] < blocksize[j]
    indexplaced = j;
}
}
if (indexPlaced != 1) {
allocation [i] = indexplaced;
occupied [indexplaced] = 1;
blocksize [indexplaced] = processsize[i];
}}
```

```
printf("\n Process No \t Process Size \t Block no \t");
for(i=0; i<process; i++){
  printf("\n %d \t\t %d \t\t", i+1, process[i]);
  if(allocation[i] != -1)
    printf("%d", allocation[i]+1);
  else
    printf("not allocated");
}
}
int main(){
  int blocks, processes;
  printf("Enter the no of blocks:");
  scanf("%d", &blocks);
  printf("Enter size of each block:");
  for(i=0; i<blocks; i++)
    scanf("%d", &processes);
  int processSize[process];
  printf("\n Enter size of each process:");
  for(i=0; i<process; i++)
    scanf("%d", processSize[i]);
  worstfit(blockSize, blocks, processSize, process);
  return 0;
}
```

Output:

Enter no of blocks: 3
Enter size of each block: 5 2 7
Enter no of process: 2
Enter size of each process: 1 4

Process No Process Size Block no

1 3
2 4
3

First fit:

```
#include <stdio.h>
#include <conio.h>
#define max 25.
void main(){
int frag[max], b[max], f[max], i, j, nb, nf,
static int bf[max], ff[max];
printf("\n\t Memory Management Scheme - first fit");
printf("Enter the no of blocks:");
scanf("%d", &nb);
printf("Enter the no of files:");
scanf("%d", &nf);
printf("Enter the size of the blocks:\n");
for(i=1; i<=nb; i++){
printf("Block %d:");
scanf("%d", &b[i]);
}
printf("\n Enter the size of the blocks:\n");
for(i=1; i<=nb; i++){
printf("Block %d:");
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]);
}
```

```
printf("\n file_no:\t file_size:\t Block_no:\t
for(i=1; i<=nf; i++){
 int allocated=0;
 for(j=1; j<=nb; j++){
   if(temp>=0){
     ff[i]=j;
      bf[j]=1;
     frag[i]=b[j]-f[i];
     allocated=1;
     printf("\n %d \t\t %d \t\t %d",
     f[i], ff[i], bf[i]);
     break;
   }
 }
}
if(!allocated){
  printf("\n %d \t\t %d \t\t Not Allocated
    \t\t -", i, f[i]);
  }
}
getch();
}
```

Output

file no	file size	Block_no	Block-size
1	1	1	5
2	4	3	4

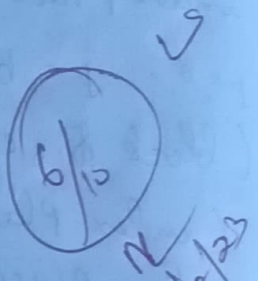
```
#include <
#define M
Void Best f
processSize[
int allocat
int occup
for(i=0;
allocat
}
for(i=0;
occupied[
}
for(i=0;
occupied[i
}
for(i=0;
int index
for(j=0;
if(block
if(ind
index
else if(
inde
}
}
int main(
int p
printf(
scanf(
int f
```

```
c) #include <stdio.h>
#define MAX 10
Void BestFit (int blockSize[], int blocks, int
processSize[], int process, int m){
int allocation[process];
int occupied [block]
for(i=0; i<process; i++){
allocation[i] = -1;

{
for(i=0; i<block; i++){
occupied[i] = 0;

{
for(i=0; i<process; i++){
occupied[i]=0;

{
for(i=0; i<process; i++){
int indexPlaced = -1;
for(j=0; j<blocks; j++){
if(blockSize[j] >= processSize[i] && !occupied[j]){
if(indexPlaced == -1)
indexPlaced = j;
else if(blockSize[j] < blockSize(indexplaced)
indexPlaced = j;

{
int main()
int p, m, j;
printf("Enter no of process & blocks:");
scanf("%d %d", &p &m);
int processSize[p], blockSize[m];
```

```
printf("Enter the Block sizes:");
for(j=0;j<m;j++)
  scanf("%d", blocksize[j]);
int process_size = sizeof(process);
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]] = flag[i];
}
}
```

Result:

Enter the no of blocks: 3
Enter the no of files: 2
Enter the size of blocks:
Block 1: 5
Block 2: 2
Block 3: 7



Enter the size of files:
file 1: 1
file 2: 4

file no	file size	Block no	Block size	fragment
1	1	2	2	1
2	4	1	5	

To simulate fa
) f If 0
include <st
define fram
Void fifo (int
int frames
int front =
for (i=0; i
frame [i]
int page fau
for (i=0;
int page
int found=
for (j=0; j
if (frame
found = 1
break;
}}
if (!found)
frames [fron
front = (fr
page fa
printf ("pa
for (j=0
if (fra
print
} else {
printf ("
}}
printf ("
printf ("T
}