Ex. No.: 10a)
Date: 09/04/25

BEST FIT

Aim:

To implement Best Fit memory allocation technique using Python.

Algorithm:

1. Input memory blocks and processes with sizes
2. Initialize all memory blocks as free.
3. Start by picking each process and find the minimum block size that can be assigned to current process
4. If found then assign it to the current process.
5. If not found then leave that process and keep checking the further processes.

Program Code:

```
#include <stdio.h>
Void bestfit(int blocks[], int, int processes[], int n){
  int allocation[n];
  for(int i=0; i<n; i++) allocation[i]=-1;
  for(int i=0; i<n; i++){
    int bestIdx=-1;
    for(int j=0; j<m; j++){
      if(blocks[j]>=processes[i]&&allocation[j]==-1){
      if(bestIdx==-1||blocks[bestIdx]>blocks[j]){
                bestIdx=j;
            }
        }
    }
    if(bestIdx!=-1){
    allocation[i]=bestIdx;
    blocks[bestIdx]-=processes[i];
```

```
Printf("Process %d allocated to block %d (size %d)\n", i+1,
        best_idx + 1, blocks[best_idx]);
    else {
      printf("process %d could not be allocated\n", i+1);
      }
    }
}

int main(){
  int m, n
  printf("Enter the number of memory blocks: ");
  scanf("%d", &m);
  printf("Enter the number of processes = ");
  scanf("%d", &n);
  int blocks[m], process[n];
  printf("Enter the sizes of %d memory blocks:\n", m);
  for(int i=0; i<m; i++){
     scanf("%d", &blocks[i]);
  }
  printf("Enter the sizes of %d processes:\n", n);
  for(int i=0; i<n; i++){
      scanf("%d", &process[i]);
  }
  best_fit(blocks, m, process, n);
  return 0;
}
```

Input:

Enter the number of memory blocks: 5

Enter the number of processes: 4

Enter the sizes of 5 memory blocks:

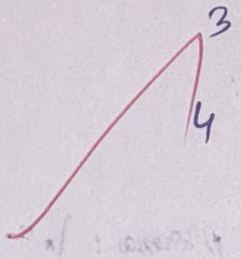
10 20 30 40 50

Enter the sizes of 4 processes

5 10 20 25

Output:

Process No	process Size	Block No.
1	5	1
2	10	2
3	10	
4	15	



Sample Output:

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5

Result:

Thus, Best fit memory allocation technique executed Successfully.

Ex. No.: 10b)
Date: 10/4/25

FIRST FIT

Aim:

To write a C program for implementation memory allocation methods for fixed partition using first fit.

Algorithm:

1. Define the max as 25.
2: Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max]. 3: Get the number of blocks,files,size of the blocks using for loop.
4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]
5: Check highest

Program Code:

```
#include <stdio.h>
#include Max 25
Void firstfit(int b[], int f[], int nb, int nf){
   int frag[MAX], bf[MAX], ff[MAX]
   int i,j;
   it highat=0;
   for(i=0;i<nb;i++){
      bf[i]=0;
   }
   for(i=0;i<nf;i++){
   for(j=0;j<nb;j++){
   if(bf[j]==0&& b[j]>=f[i]){
      ff[i]=j;
frag[i]=b[j]-f[i];
```

```
            bf[j] = 1;
            highest = 1;
            break;
        }
    }
    if (highest == 0){
        ff[i] = -1;
        frag[i] = -1;
    }
    highest = 0;
}
printf("\n file no. \t file size \t Block No. \t Block Size \t fragmentation \n");
for (i=0; i<n; i++){
    printf("%d \t\t %d \t\t %d \t\t", i+1, f[i]);
    if (ff[i] != -1){
        printf("%d \t\t %d \t\t", i+1, b[ff[i]]);
        if (ff[i] != -1){
            printf("%d \t\t %d \t\t %d \n", ff[i]+1,
                                b[ff[i]] - frag[i]);
        }
        else{
            printf("Not Allocated \t\t \t\t \n");
        }
    }
}
```

```
int main(){
  int b[max], f[MAX];
  int nb, nf;
  int i;
printf("Enter the number of memory blocks: ");
scanf("%d", &nb);
printf("Enter the number of files: ");
scanf("%d", &nf);
printf("\n Enter the Size of blocks: \n");
for(i=0; i<nb; i++){ printf("Block-%d: ", i+1);
                     scanf("%d", &b[i]);
}

printf("\n Enter the Sizes of the files: \n");
for(i=0; i<nf; i++){
    printf("file %d: ", i+1);
    scanf("%d", &f[i]);
    }
    firstfit(b, nb, nf);
    return 0;
}
```

Input:

Enter the number of memory blocks: 5

Enter the number of files: 4

Enter the Size of the blocks:

Block 1: 100

Block 2: 200

Block 3: 300

Block 4: 150

Block 5: 500

Enter the Size of the file:

file 1: 120

file 2: 300

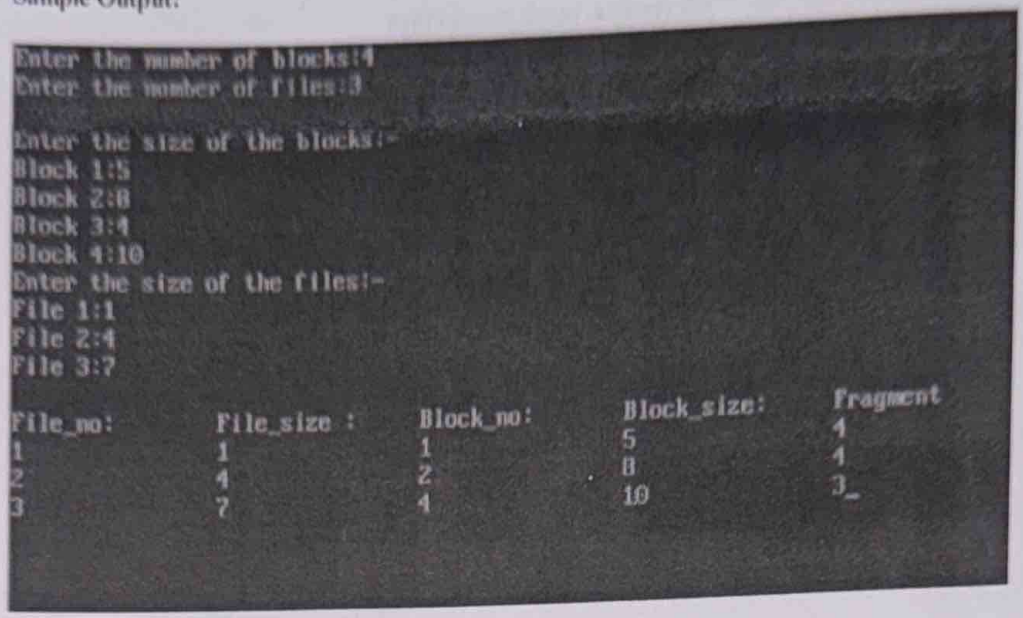
file 3: 150

file 4: 200

Output:

file No.	file Size	Block No	Block Size	fragmentation
1	120	2	100	80
2	300	3	300	0
3	150	4	150	0
4	200	5	200	300

Sample Output:



Result:

Thus, memory allocation methods for fixed partition using first fit executed successfully.