Ex. No.: 10a)
Date: 9/4/25

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:

```
def bestfit(bsize, m, psize, n):
    alloc = [-1]*n
    for i in range(n):
        bestidx = -1
        for j in range(m):
            if bsize[j] >= psize[i]:
                if bestidx == -1:
                    bestidx = j
            elif bsize[bestidx] > bsize[j]:
                bestidx = j
        if bestIdx != -1:
            alloc[i] = bestidx
            bsize[bestidx] -= psize[i]
for i in range(n):
    print(i+1," ", psize[i], end=" ")
    if(alloc[i] != -1):
        print(alloc[i]+1)
```

```
else:
    print("Not Allocated");

if __name__ == '__main__':
    bsize = [100, 500, 200, 300, 600]
    psize = [212, 417, 312, 426]
    m = len(bsize)
    n = len(psize)
    bestfit(bsize, m, psize, n)
```

Output:-

Process No	Process Size	Blocks No
1	212	4
2	417	2
3	312	5
4	426	Not Allocated

Sample Output:

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

Result:

Hence the Best Fit for the given processes is Implemented and Verified

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>
#define MAX 25
int main()
{
    int bs[Max], bo[MAX], bc, fs[Max], fc,
        alloc[MAX], fragment[MAX];
    printf("Enter no of Memory Blocks (max
        %d): ", Max);
    scanf("%d", &bc);
    printf("Enter size of Each blocks: \n");
    for (int i=0; i<bc; i++)
    {
        printf("Block %d size: ", i+1);
        scanf("%d", &bs[i]);
        bo[i] = bs[i];
    }
```

```
printf("\n Enter no of files (max %d):",
                                    MAX);
scanf("%d", &fc);
printf("Enter size of each file:\n");
for (int i=0; i<fc; i++)
{   printf("File %d size", i+1);
    scanf("%d", &fs[i]);
    alloc[i]= -1;
    fragment[i] =0;

}
for (int i=0; i<fc; i++)

{   for (int j=0; j<bc; j++){
        if (bs[j] >= fs[i])
        {
            alloc[i] = j;
            fragment[i] = bs[j] - fs[i];
            bs[j] - = fs[i]
            break;

        }

    }

}
printf("\n File No \t File size \t Block
        No \t Block size \t Fragment \n")
for (int i=0; i<fc; i++)
{
    printf("%d \t\t %d \t\t", i+1, fs[i]);
    if (alloc[i] != -1)
    {
        int bno=alloc[i];
        printf("%d \t\t %d \t\t %d \n",
                bno+1, bs[bno], fragment[i])
    }
```

```
    else{
        printf("Not Allocated\t-\t\t-\n");
    }
  }
  return0;
}
```

Output :-

Enter no of memory Blocks :- 4

Block 1 size : 300

Block 2 size : 250

Block 3 size : 400

Block 4 size : 500

Enter no of files : 4

File 1 size : 212

File 2 size : 200

File 3 size : 312

File 4 size : 417

File No	File size	Block No	Block size	Fragmen
1	212	2	250	38
2	200	1	300	100
3	312	3	400	88
4	417	4	500	83

Sample Output:

```
Enter the number of blocks:4
Enter the number of files:3

Enter the size of the blocks:-
Block 1:5
Block 2:8
Block 3:1
Block 4:10
Enter the size of the files:-
File 1:1
File 2:4
File 3:7

File_no:        File_size :     Block_no:       Block_size:     Fragment
1               1               1               5               4
2               4               2               8               4
3               7               4               10              3
```

Result:

Hence the first fit algorithm is Implemented and executed.