Ex. No.: 10a)
Date: 12/4/25

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:

```c
#include <stdio.h>
#define max 25

int main(){
    int frag[max], b[max], f[max], i, j, nb, nf, bfIndex;
    static int bf[max], ff[max];

    printf("Enter number of blocks: ");
    scanf("%d", &nb);

    printf("Enter number of files: ");
    scanf("%d", &nf);

    printf("\nEnter size of each block:\n");
    for(i=0; i<nb; i++){
        printf("Block %d: ", i+1);
        scanf("%d", &b[i]);
    }

    printf("\nEnter size of each file:\n");
    for(i=0; i<nf; i++){
        printf("File %d: ", i+1);
        scanf("%d", &f[i]);
    }

    for(i=0; i<nf; i++){
        int bestFit = -1;
        for(j=0; j<nb; j++){
            if(bf[j]==0 && b[j]>=f[i]){
                if(bestFit==-1 || b[j]<b[bestFit]){
                    bestFit = j;
                }
            }
        }
        if(bestFit != -1){
            ff[i] = bestFit;
            frag[i] = b[bestFit]-f[i];
            bf[bestFit] = 1;
        } else {
```

```c
            ff[i] = -1;
            frag[i] = -1;
        }
    }
    printf("\nFile No\t File Size\t Block No\t Block Size\t Fragment \n");
    for (i = 0; i < nf; i++) {
        if (ff[i] != -1) {
            printf("%d\t %d \t %d\t %d\t %d\n",
                i+1, f[i], ff[i]+1, b[ff[i]], frag[i]);
        } else {
            printf("%d\t %d \t\t Not Allocated \t-\t-\n",
                i+1, f[i]);
        }
    }
    return 0;
}
```

Sample Output:

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

Result:

C program for implementation of Best Fit memory allocation has been executed successfully.