Ex. No.: 10A **Roll no:231901004**

Date: ***2.4.2025***

BEST FIT

Aim:

To implement the Best Fit memory allocation technique using Python.

Algorithm:

1. Input memory blocks and processes with their sizes.
2. Initialize all memory blocks as free.
3. For each process, find the smallest memory block that can accommodate it.
4. If such a block is found, allocate it to the process.
5. If no suitable block is found, leave the process unallocated.

Program Code (best_fit.py):

```
def best_fit(blockSize, processSize):
allocation = [-1] * len(processSize)

   for i in range(len(processSize)):
       best_idx = -1        for j in range(len(blockSize)):            if
blockSize[j] >= processSize[i]:                if best_idx == -1 or
blockSize[j] < blockSize[best_idx]:
                 best_idx = j
if best_idx != -1:
           allocation[i] = best_idx + 1
           blockSize[best_idx] -= processSize[i]

   print("Process No.\tProcess Size\tBlock No.")
for i in range(len(processSize)):
        print(f"{i + 1}\t\t{processSize[i]}\t\t", end="")
        if allocation[i] != -1:
            print(f"{allocation[i]}")
        else:
            print("Not Allocated")


# Example usage blockSize = [100,
500, 200, 300, 600] processSize =
[212, 417, 112, 426]


best_fit(blockSize, processSize)
```

Sample Output:

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

Result:

Thus, the Best Fit memory allocation technique was successfully implemented in Python.