Ex. No.: 10a)

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 best_fit(blocks, processes):
    allocation = [-1] * len(processes)
    for i in range(len(processes)):
        best_idx = -1
        min_size = float('inf')
        for j in range(len(blocks)):
            if blocks[j] >= processes[i] and blocks[j] - processes[i] < min_size:
                best_idx = j
                min_size = blocks[j] - processes[i]
        if best_idx != -1:
            allocation[i] = best_idx
            blocks[best_idx] -= processes[i]
    print("\nProcess No.\tProcess Size\tBlock No.\tBlock Size")
    for i in range(len(processes)):
        if allocation[i] != -1:
            print(f"{i+1}\t\t{processes[i]}\t\t{allocation[i]+1}\t\t{blocks[allocation[i]] + processes[i]}")
        else:
            print(f"{i+1}\t\t{processes[i]}\t\tNot Allocated")

if __name__ == "__main__":
    blocks = [100, 500, 200, 300, 600]
    processes = [212, 417, 112, 426]
    best_fit(blocks, processes)
```

Sample Output:

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

Output:

Process No.	Process Size	Block No.	Block Size
1	212	1	100
2	417	3	200
3	112	4	300
4	426	5	600

Result:

Best Fit memory allocation technique has been successfully implemented and the output has been verified.