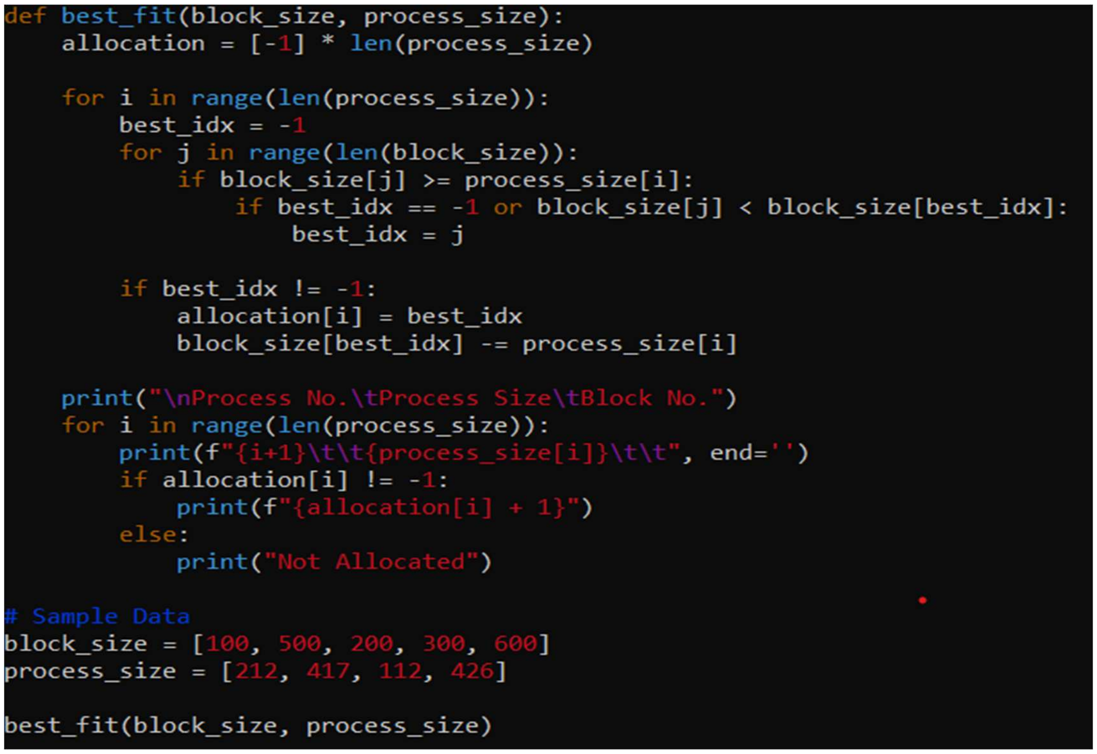
Kamalesh S P
230701138
II-CSE-B

CS23431-Operating System

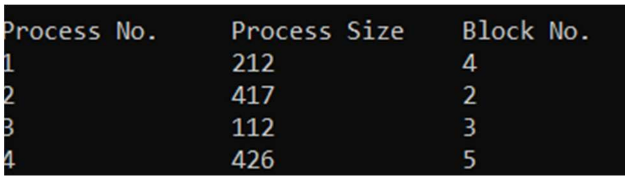
10a) Best Fit memory allocation technique

Program Code:



```
def best_fit(block_size, process_size):
    allocation = [-1] * len(process_size)

    for i in range(len(process_size)):
        best_idx = -1
        for j in range(len(block_size)):
            if block_size[j] >= process_size[i]:
                if best_idx == -1 or block_size[j] < block_size[best_idx]:
                    best_idx = j

        if best_idx != -1:
            allocation[i] = best_idx
            block_size[best_idx] -= process_size[i]

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

# Sample Data
block_size = [100, 500, 200, 300, 600]
process_size = [212, 417, 112, 426]

best_fit(block_size, process_size)
```

Output:

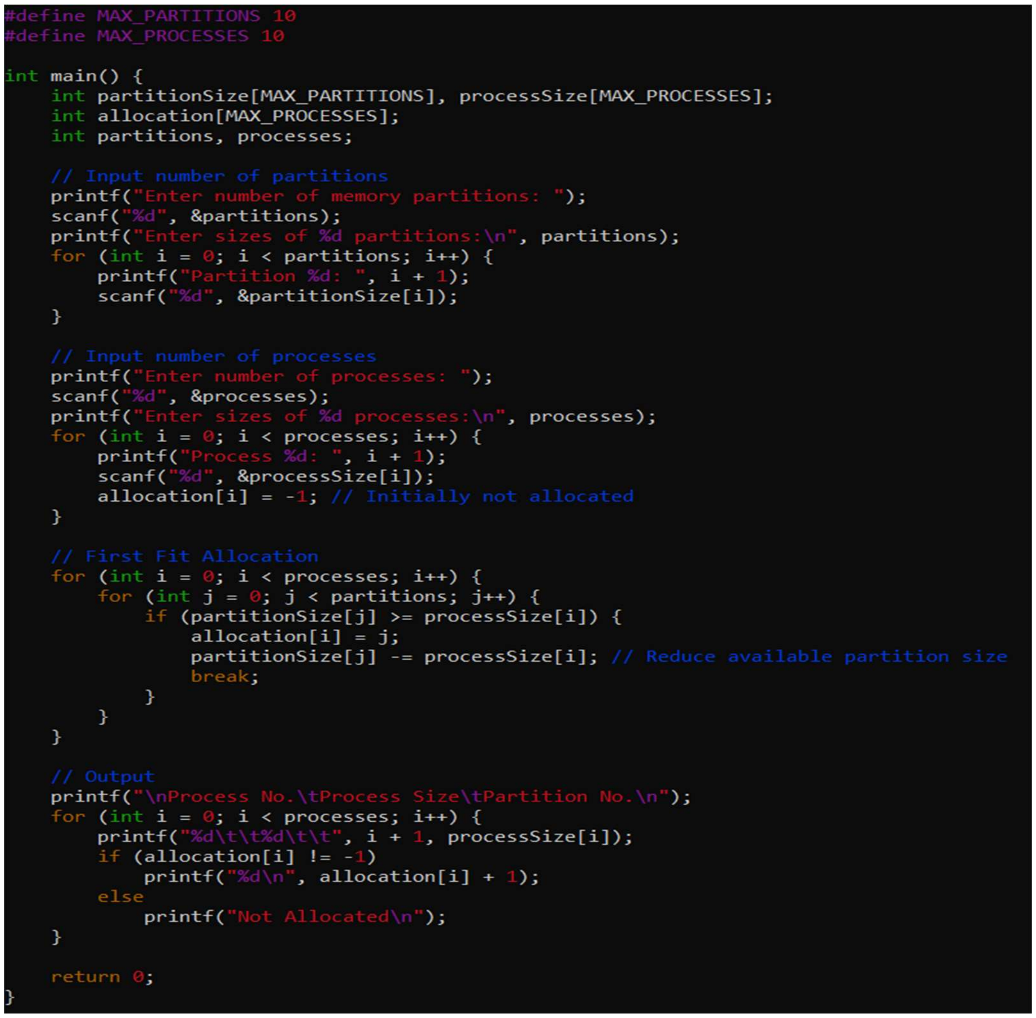


```
Process No.     Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5
```

Kamalesh S P
230701138
II-CSE-B

10b) memory allocation methods for fixed partition using first fit

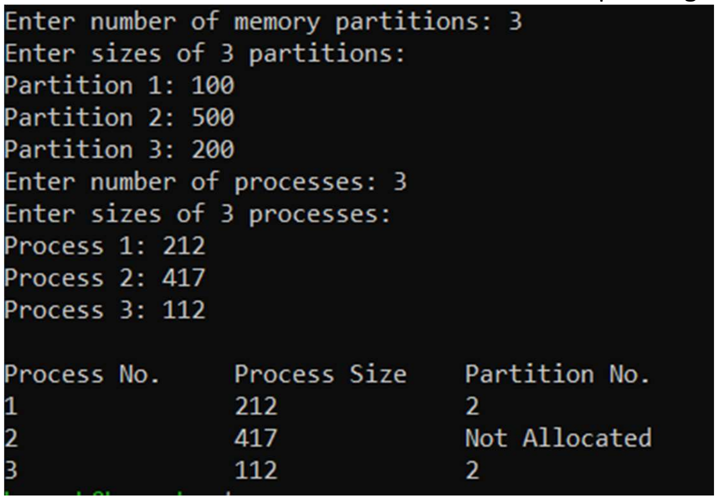
Program Code:



```
#define MAX_PARTITIONS 10
#define MAX_PROCESSES 10

int main() {
    int partitionSize[MAX_PARTITIONS], processSize[MAX_PROCESSES];
    int allocation[MAX_PROCESSES];
    int partitions, processes;

    // Input number of partitions
    printf("Enter number of memory partitions: ");
    scanf("%d", &partitions);
    printf("Enter sizes of %d partitions:\n", partitions);
    for (int i = 0; i < partitions; i++) {
        printf("Partition %d: ", i + 1);
        scanf("%d", &partitionSize[i]);
    }

    // Input number of processes
    printf("Enter number of processes: ");
    scanf("%d", &processes);
    printf("Enter sizes of %d processes:\n", processes);
    for (int i = 0; i < processes; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &processSize[i]);
        allocation[i] = -1; // Initially not allocated
    }

    // First Fit Allocation
    for (int i = 0; i < processes; i++) {
        for (int j = 0; j < partitions; j++) {
            if (partitionSize[j] >= processSize[i]) {
                allocation[i] = j;
                partitionSize[j] -= processSize[i]; // Reduce available partition size
                break;
            }
        }
    }

    // Output
    printf("\nProcess No.\tProcess Size\tPartition No.\n");
    for (int i = 0; i < processes; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }

    return 0;
}
```

Output:

CS23431-Operating System



```
Enter number of memory partitions: 3
Enter sizes of 3 partitions:
Partition 1: 100
Partition 2: 500
Partition 3: 200
Enter number of processes: 3
Enter sizes of 3 processes:
Process 1: 212
Process 2: 417
Process 3: 112

Process No.     Process Size    Partition No.
1               212             2
2               417             Not Allocated
3               112             2
```