Assignment - 5

OPERATING SYSTEM

Topic: Memory Management

Q: Write a C program to simulate the MVT and MFT memory management techniques.

DESCRIPTION:

MFT (Multiprogramming with a Fixed Number of Tasks) is one of the old memory management techniques in which the memory is partitioned into fixed-size partitions and each job is assigned to a partition. The memory assigned to a partition does not change. *MVT (Multiprogramming with a Variable Number of Tasks)* is a memory management technique in which each job gets just the amount of memory it needs. That is, the partitioning of memory is dynamic and changes as jobs enter and leave the system. MVT is a more ``efficient'' user of resources. MFT suffers from the problem of internal fragmentation, and MVT suffers from external fragmentation.

MFT MEMORY MANAGEMENT TECHNIQUE

SAMPLE INPUT:

```
Enter the total memory available (in Bytes) -- 1000
Enter the block size (in Bytes)-- 300
Enter the number of processes – 5
Enter memory required for process 1 (in Bytes) -- 275
Enter memory required for process 2 (in Bytes) -- 400
Enter memory required for process 3 (in Bytes) -- 290
Enter memory required for process 4 (in Bytes) -- 293
Enter memory required for process 5 (in Bytes) -- 100

No. of Blocks available in memory -- 3
```

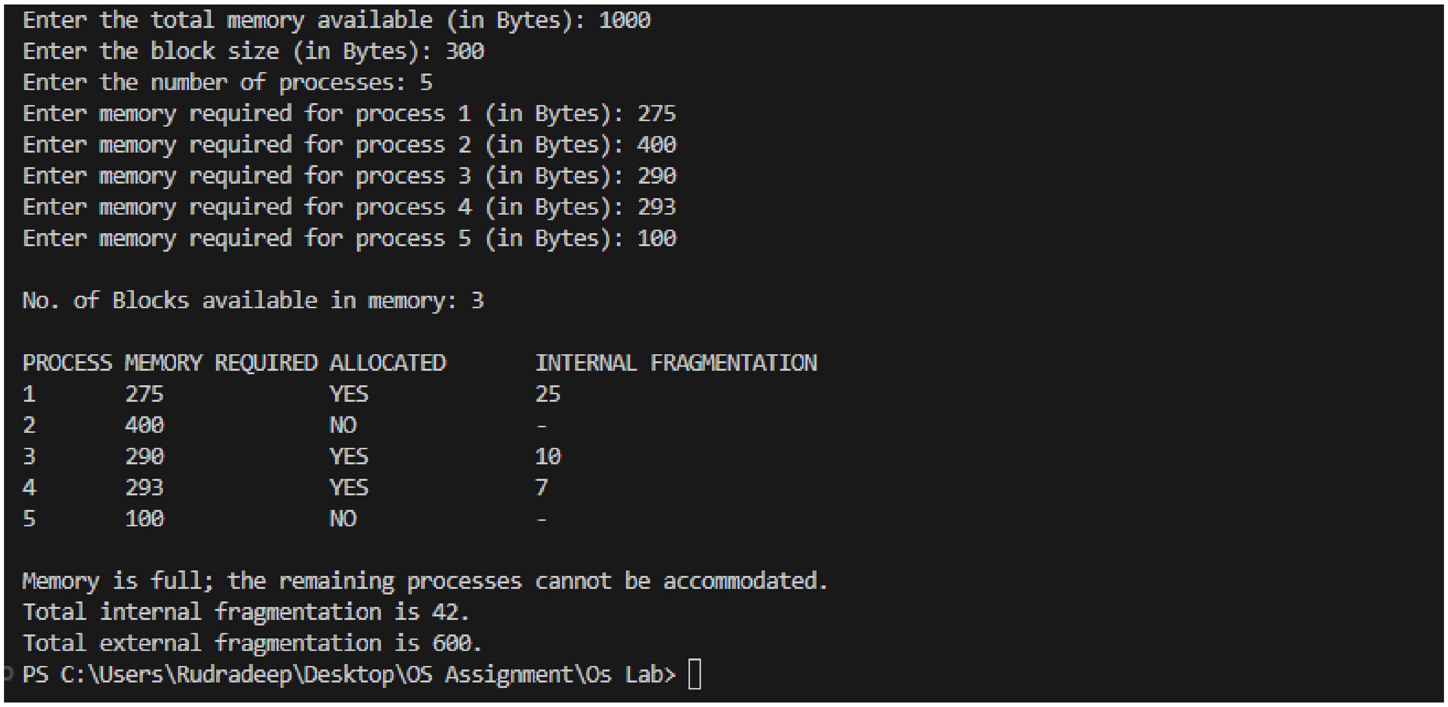
SAMPLE OUTPUT:

PROCESS	MEMORY REQUIRED	ALLOCATED	INTERNAL FRAGMENTATION

1	275	YES	25
2	400	NO	—
3	290	YES	10
4	293	YES	7

Memory is full; the remaining processes cannot be accommodated.
The total internal fragmentation is 42.
Total External Fragmentation is 100

```c
#include <stdio.h>

int main() {
    int total_memory, block_size, num_processes;
    int internal_fragmentation = 0, external_fragmentation = 0;
    int memory_used = 0;

    printf("Enter the total memory available (in Bytes): ");
    scanf("%d", &total_memory);

    printf("Enter the block size (in Bytes): ");
    scanf("%d", &block_size);

    printf("Enter the number of processes: ");
    scanf("%d", &num_processes);

    int memory_required[num_processes];
    for (int i = 0; i < num_processes; i++) {
        printf("Enter memory required for process %d (in Bytes): ", i + 1);
        scanf("%d", &memory_required[i]);
    }

    int num_blocks = total_memory / block_size;
    int remaining_memory = total_memory - (num_blocks * block_size);
    printf("\nNo. of Blocks available in memory: %d\n\n", num_blocks);
    printf("PROCESS\tMEMORY REQUIRED\tALLOCATED\tINTERNAL FRAGMENTATION\n");

    for (int i = 0; i < num_processes; i++) {
        if (num_blocks > 0 && memory_required[i] <= block_size) {
            printf("%d\t%d\t\tYES\t\t%d\n", i + 1, memory_required[i], block_size -
memory_required[i]);
            internal_fragmentation += block_size - memory_required[i];
            num_blocks--;
            memory_used += memory_required[i];
        } else {
            printf("%d\t%d\t\tNO\t\t-\n", i + 1, memory_required[i]);
            external_fragmentation += memory_required[i];
        }
    }

    printf("\nMemory is full; the remaining processes cannot be accommodated.\n");
    printf("Total internal fragmentation is %d.\n", internal_fragmentation);
    printf("Total external fragmentation is %d.\n", remaining_memory +
external_fragmentation);

    return 0;
}
```



```
Enter the total memory available (in Bytes): 1000
Enter the block size (in Bytes): 300
Enter the number of processes: 5
Enter memory required for process 1 (in Bytes): 275
Enter memory required for process 2 (in Bytes): 400
Enter memory required for process 3 (in Bytes): 290
Enter memory required for process 4 (in Bytes): 293
Enter memory required for process 5 (in Bytes): 100

No. of Blocks available in memory: 3

PROCESS MEMORY REQUIRED ALLOCATED       INTERNAL FRAGMENTATION
1       275             YES             25
2       400             NO              -
3       290             YES             10
4       293             YES             7
5       100             NO              -

Memory is full; the remaining processes cannot be accommodated.
Total internal fragmentation is 42.
Total external fragmentation is 600.
PS C:\Users\Rudradeep\Desktop\OS Assignment\Os Lab> 
```

MVT MEMORY MANAGEMENT TECHNIQUE

SAMPLE INPUT:

Enter the total memory available (in Bytes) -- 1000

Enter the memory required for process 1 (in Bytes) -- 400
Memory is allocated for Process 1
Do you want to continue(y/n) -- y

Enter memory required for process 2 (in Bytes) -- 275
Memory is allocated for Process 2
Do you want to continue(y/n) -- y

Enter memory required for process 3 (in Bytes) -- 550

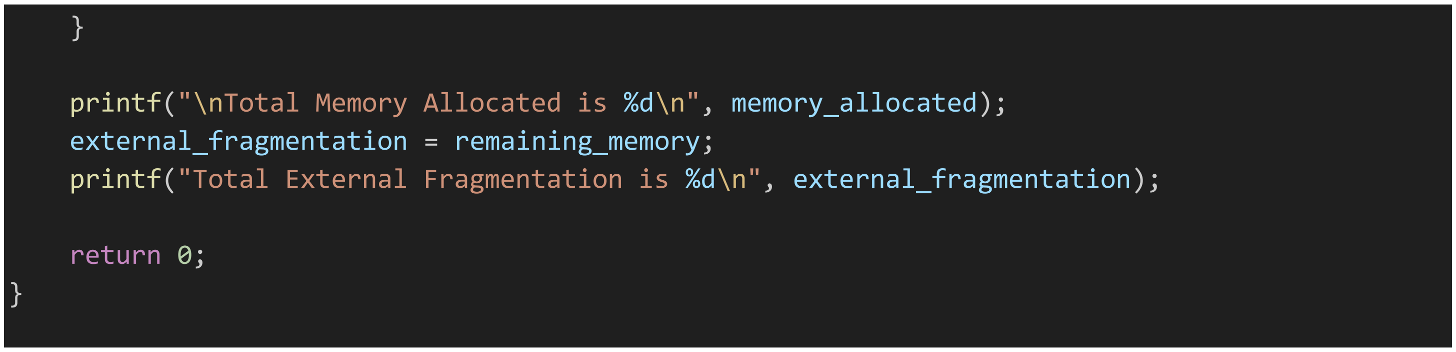
SAMPLE OUTPUT:

Memory is Full
Total Memory Available -- 1000

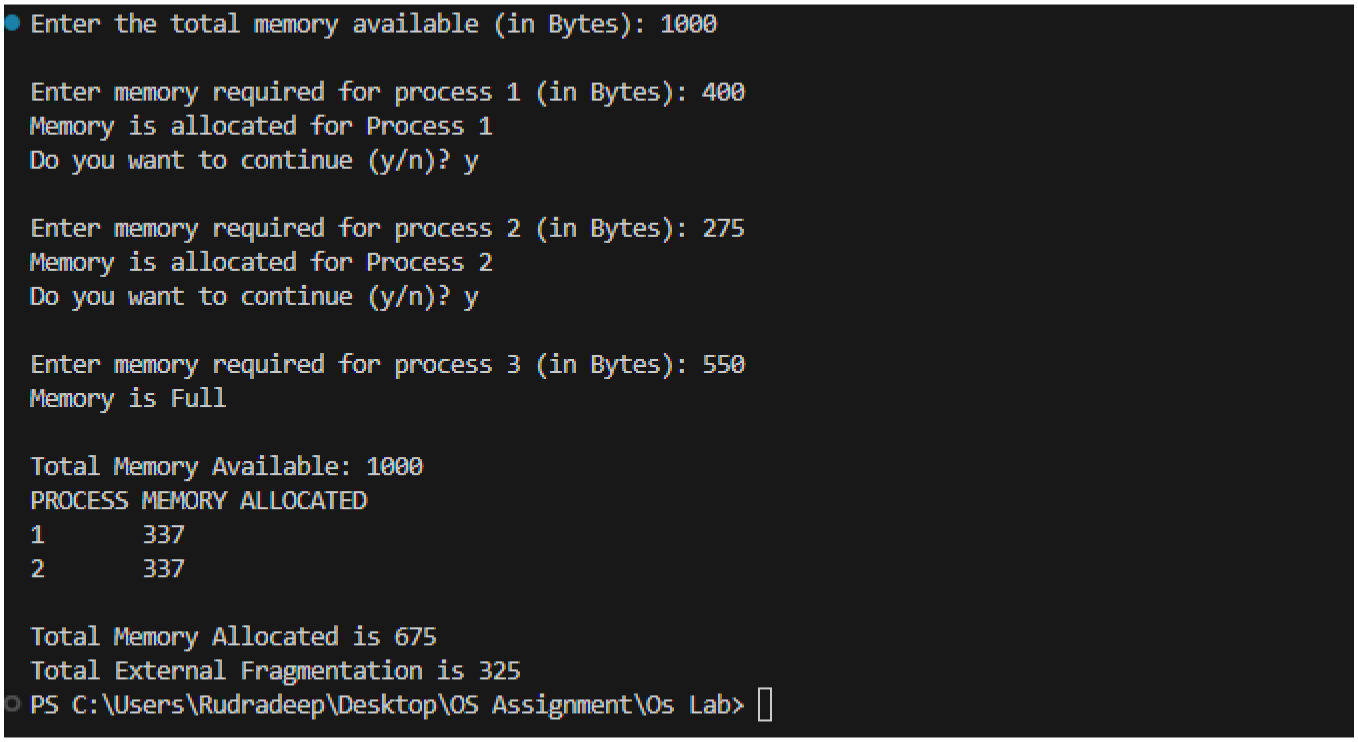
PROCESS	MEMORY ALLOCATED
1	400
2	275

Total Memory Allocated is 675
Total External Fragmentation is 325

```c
#include <stdio.h>

int main() {
    int total_memory, memory_allocated = 0, external_fragmentation = 0;
    int memory_required, process_count = 0;
    char choice;

    printf("Enter the total memory available (in Bytes): ");
    scanf("%d", &total_memory);

    int remaining_memory = total_memory;

    do {
        printf("\nEnter memory required for process %d (in Bytes): ", process_count + 1);
        scanf("%d", &memory_required);

        if (memory_required <= remaining_memory) {
            printf("Memory is allocated for Process %d\n", process_count + 1);
            memory_allocated += memory_required;
            remaining_memory -= memory_required;
            process_count++;
        } else {
            printf("Memory is Full\n");
            break;
        }

        printf("Do you want to continue (y/n)? ");
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');

    printf("\nTotal Memory Available: %d\n", total_memory);
    printf("PROCESS\tMEMORY ALLOCATED\n");

    for (int i = 1; i <= process_count; i++) {
        printf("%d\t%d\n", i, memory_allocated / process_count);  // For simplicity, evenly distributed among allocated processes
```



```c
    }

    printf("\nTotal Memory Allocated is %d\n", memory_allocated);
    external_fragmentation = remaining_memory;
    printf("Total External Fragmentation is %d\n", external_fragmentation);

    return 0;
}
```



```
Enter the total memory available (in Bytes): 1000

Enter memory required for process 1 (in Bytes): 400
Memory is allocated for Process 1
Do you want to continue (y/n)? y

Enter memory required for process 2 (in Bytes): 275
Memory is allocated for Process 2
Do you want to continue (y/n)? y

Enter memory required for process 3 (in Bytes): 550
Memory is Full

Total Memory Available: 1000
PROCESS MEMORY ALLOCATED
1       337
2       337

Total Memory Allocated is 675
Total External Fragmentation is 325
PS C:\Users\Rudradeep\Desktop\OS Assignment\Os Lab> 
```