Assignment - 05
TOPIC: Memory Management

---

1. **Write a C program to simulate the MVT and MFT memory management techniques.**

   ->Ans:

```
#include <stdio.h>

// Function for MFT memory management technique
void MFT() {
    int total_memory, block_size, num_blocks, num_processes, i, internal_frag
= 0, external_frag;
    int memory_required[10], allocated[10];

    printf("MFT MEMORY MANAGEMENT TECHNIQUE\n");
    printf("Enter the total memory available (in Bytes): ");
    scanf("%d", &total_memory);

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

    num_blocks = total_memory / block_size;
    external_frag = total_memory - (num_blocks * block_size);

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

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

        if (memory_required[i] <= block_size && num_blocks > 0) {
            allocated[i] = 1;
            internal_frag += block_size - memory_required[i];
            num_blocks--;
        } else {
            allocated[i] = 0;
        }
    }

    printf("\nNo. of Blocks available in memory: %d\n", total_memory /
block_size);
    printf("\nPROCESS\tMEMORY REQUIRED\tALLOCATED\tINTERNAL FRAGMENTATION\n");

    for (i = 0; i < num_processes; i++) {
        printf("%d\t%d\t\t%s\t\t", i + 1, memory_required[i], allocated[i] ?
"YES" : "NO");
        if (allocated[i]) {
            printf("%d\n", block_size - memory_required[i]);
        } else {
            printf("-\n");
        }
    }

    if (num_blocks == 0) {
```

```
        printf("\nMemory is full; the remaining processes cannot be
accommodated.\n");
    }
    printf("The total internal fragmentation is %d.\n", internal_frag);
    printf("Total External Fragmentation is %d.\n", external_frag);
}

// Function for MVT memory management technique
void MVT() {
    int total_memory, memory_allocated = 0, memory_required, i = 1;
    char choice;
    int allocated_memory[10];  // Array to store allocated memory for each
process

    printf("\nMVT MEMORY MANAGEMENT TECHNIQUE\n");
    printf("Enter the total memory available (in Bytes): ");
    scanf("%d", &total_memory);

    do {
        printf("Enter memory required for process %d (in Bytes): ", i);
        scanf("%d", &memory_required);

        if (memory_required <= total_memory - memory_allocated) {
            allocated_memory[i-1] = memory_required;  // Store the allocated
memory for this process
            memory_allocated += memory_required;
            printf("Memory is allocated for Process %d\n", i);
        } else {
            printf("Memory is Full\n");
            break;
        }

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

    printf("\nPROCESS\tMEMORY ALLOCATED\n");
    for (int j = 0; j < i - 1; j++) {
        printf("%d\t%d\n", j + 1, allocated_memory[j]);
    }

    printf("\nTotal Memory Allocated is %d\n", memory_allocated);
    printf("Total External Fragmentation is %d\n", total_memory -
memory_allocated);
}

int main() {
    int choice;
    do {
        printf("\nMemory Management Simulation\n");
        printf("1. MFT\n");
        printf("2. MVT\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

```
                MFT();
                break;
            case 2:
                MVT();
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice, please try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Output:

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ vim MemoryManagement.c
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ gcc MemoryManagement.c -o MemoryManagement
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ./MemoryManagement

Memory Management Simulation
1. MFT
2. MVT
3. Exit
Enter your choice: 1
MFT MEMORY MANAGEMENT TECHNIQUE
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.
The total internal fragmentation is 42.
Total External Fragmentation is 100.
```

```
Memory Management Simulation
1. MFT
2. MVT
3. Exit
Enter your choice: 2

MVT MEMORY MANAGEMENT TECHNIQUE
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

PROCESS MEMORY ALLOCATED
1       400
2       275

Total Memory Allocated is 675
Total External Fragmentation is 325

Memory Management Simulation
1. MFT
2. MVT
3. Exit
Enter your choice: 3
Exiting...
```