LAB PROGRAM 9

Write a C program to simulate the following contiguous memory allocation techniques

a) Worst-fit

b) Best-fit

c) First-fit

INPUT

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

void printAllocation(const char *scheme, int allocation[], int processSize[], int blockSize[], int originalBlockSize[], int n) {
    printf("Memory Management Scheme - %s\n", scheme);
    printf("File_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment\n");
    for(int i = 0; i < n; i++) {
        if(allocation[i] != -1) {
            printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i+1, processSize[i], allocation[i]+1, originalBlockSize[allocation[i]], blockSize[allocation[i]]);
        } else {
            printf("%d\t\t%d\t\tNot Allocated\n", i+1, processSize[i]);
        }
    }
    printf("\n");
}

void allocateMemory(int blockSize[], int m, int processSize[], int n, const char *scheme) {
    int allocation[n], originalBlockSize[m];
    for(int i = 0; i < m; i++) {
        originalBlockSize[i] = blockSize[i];
    }

    for(int i = 0; i < n; i++) {
        allocation[i] = -1;
    }

    for(int i = 0; i < n; i++) {
        int idx = -1;
        for(int j = 0; j < m; j++) {
            if(blockSize[j] >= processSize[i]) {
                if(scheme == "First Fit" || (scheme == "Best Fit" && (idx == -1 || blockSize[j] < blockSize[idx])) ||
                   (scheme == "Worst Fit" && (idx == -1 || blockSize[j] > blockSize[idx]))) {
                    idx = j;
                    if(scheme == "First Fit") break;
                }
            }
        }

        if(idx != -1) {
            allocation[i] = idx;
            blockSize[idx] -= processSize[i];
        }
    }

    printAllocation(scheme, allocation, processSize, blockSize, originalBlockSize, n);
}

int main() {
    int blockSize[MAX], processSize[MAX];
    int m, n;

    printf("Enter the number of blocks: ");
```

```
    scanf("%d", &m);
    printf("Enter the size of each block: \n");
    for(int i = 0; i < m; i++) {
        printf("Block %d: ", i+1);
        scanf("%d", &blockSize[i]);
    }

    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the size of each process: \n");
    for(int i = 0; i < n; i++) {
        printf("Process %d: ", i+1);
        scanf("%d", &processSize[i]);
    }

    int blockSize1[MAX], blockSize2[MAX], blockSize3[MAX];
    for(int i = 0; i < m; i++) {
        blockSize1[i] = blockSize2[i] = blockSize3[i] = blockSize[i];
    }

    allocateMemory(blockSize1, m, processSize, n, "First Fit");
    allocateMemory(blockSize2, m, processSize, n, "Best Fit");
    allocateMemory(blockSize3, m, processSize, n, "Worst Fit");

    return 0;
}
```

OUTPUT

```
Enter the number of blocks: 5
Enter the size of each block:
Block 1: 400
Block 2: 700
Block 3: 200
Block 4: 300
Block 5: 600
Enter the number of processes: 4
Enter the size of each process:
Process 1: 212
Process 2: 517
Process 3: 312
Process 4: 526
Memory Management Scheme - First Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               212             1               400             188
2               517             2               700             183
3               312             5               600             288
4               526             Not Allocated

Memory Management Scheme - Best Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               212             4               300             88
2               517             5               600             83
3               312             1               400             88
4               526             2               700             174

Memory Management Scheme - Worst Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               212             2               700             176
2               517             5               600             83
3               312             2               700             176
4               526             Not Allocated


Process returned 0 (0x0)   execution time : 34.341 s
Press any key to continue.
```