LAB09:

Write a C program to simulate the following contiguous memory allocation

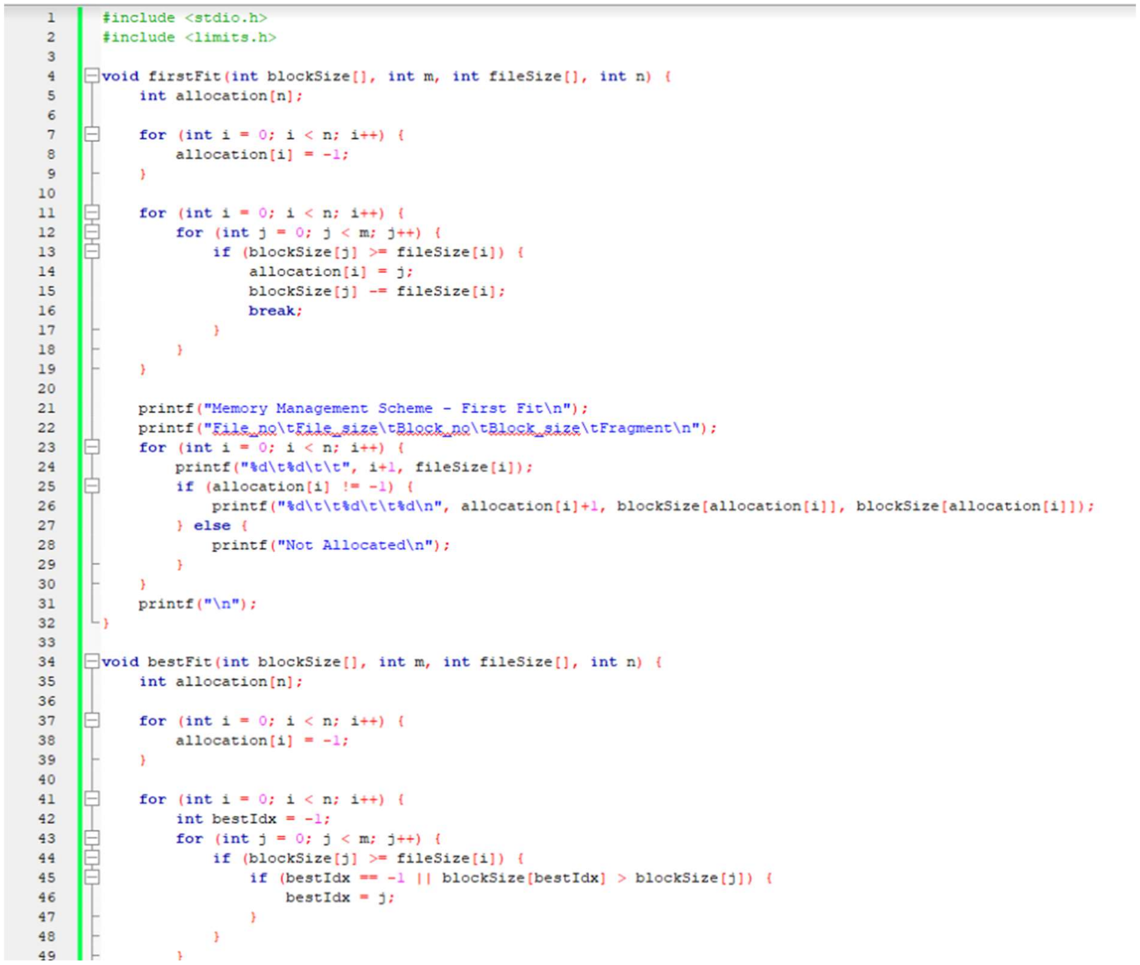
techniques

a) Worst-fit

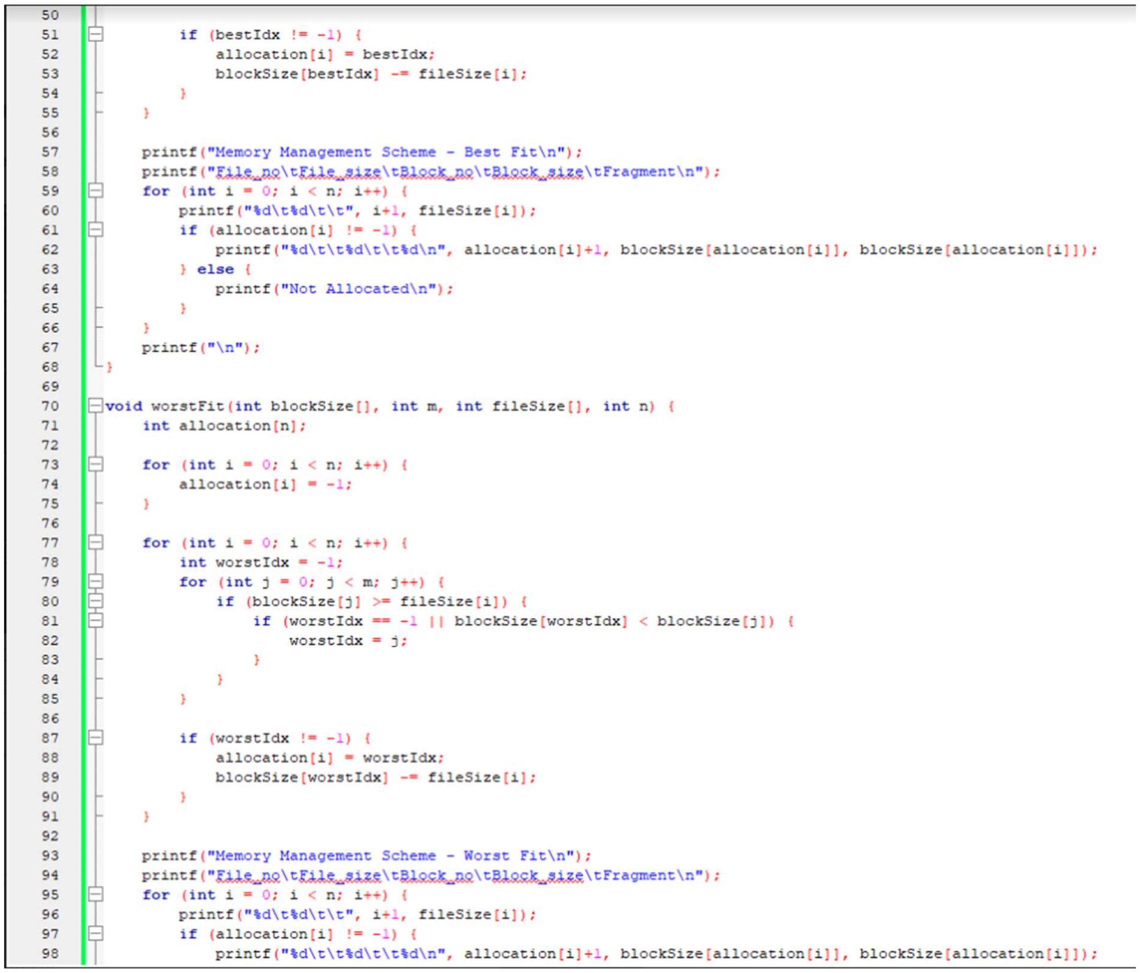
b) Best-fit

c) First-fit

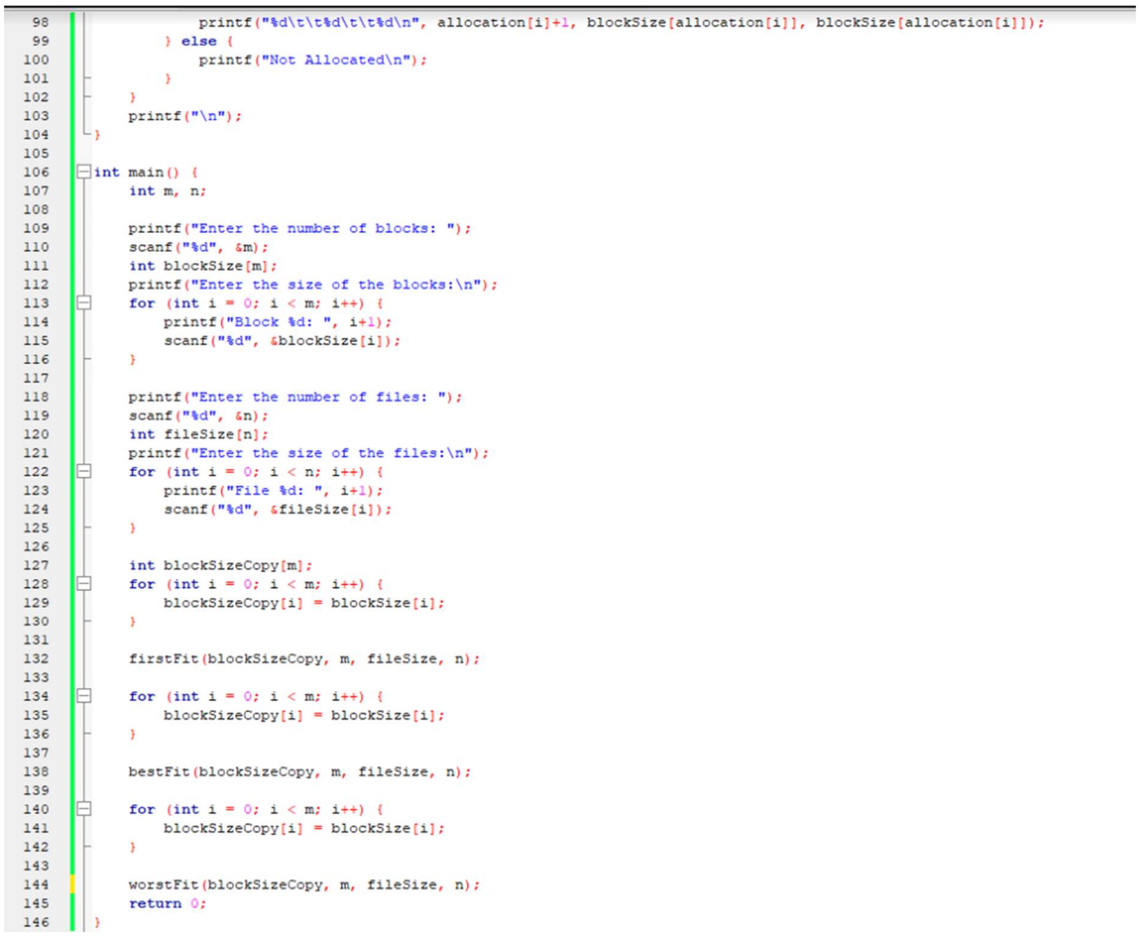
INPUT:



```c
#include <stdio.h>
#include <limits.h>

void firstFit(int blockSize[], int m, int fileSize[], int n) {
    int allocation[n];

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

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= fileSize[i]) {
                allocation[i] = j;
                blockSize[j] -= fileSize[i];
                break;
            }
        }
    }

    printf("Memory Management Scheme - First Fit\n");
    printf("File_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t", i+1, fileSize[i]);
        if (allocation[i] != -1) {
            printf("%d\t\t%d\t\t%d\n", allocation[i]+1, blockSize[allocation[i]], blockSize[allocation[i]]);
        } else {
            printf("Not Allocated\n");
        }
    }
    printf("\n");
}

void bestFit(int blockSize[], int m, int fileSize[], int n) {
    int allocation[n];

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

    for (int i = 0; i < n; i++) {
        int bestIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= fileSize[i]) {
                if (bestIdx == -1 || blockSize[bestIdx] > blockSize[j]) {
                    bestIdx = j;
                }
            }
        }
```

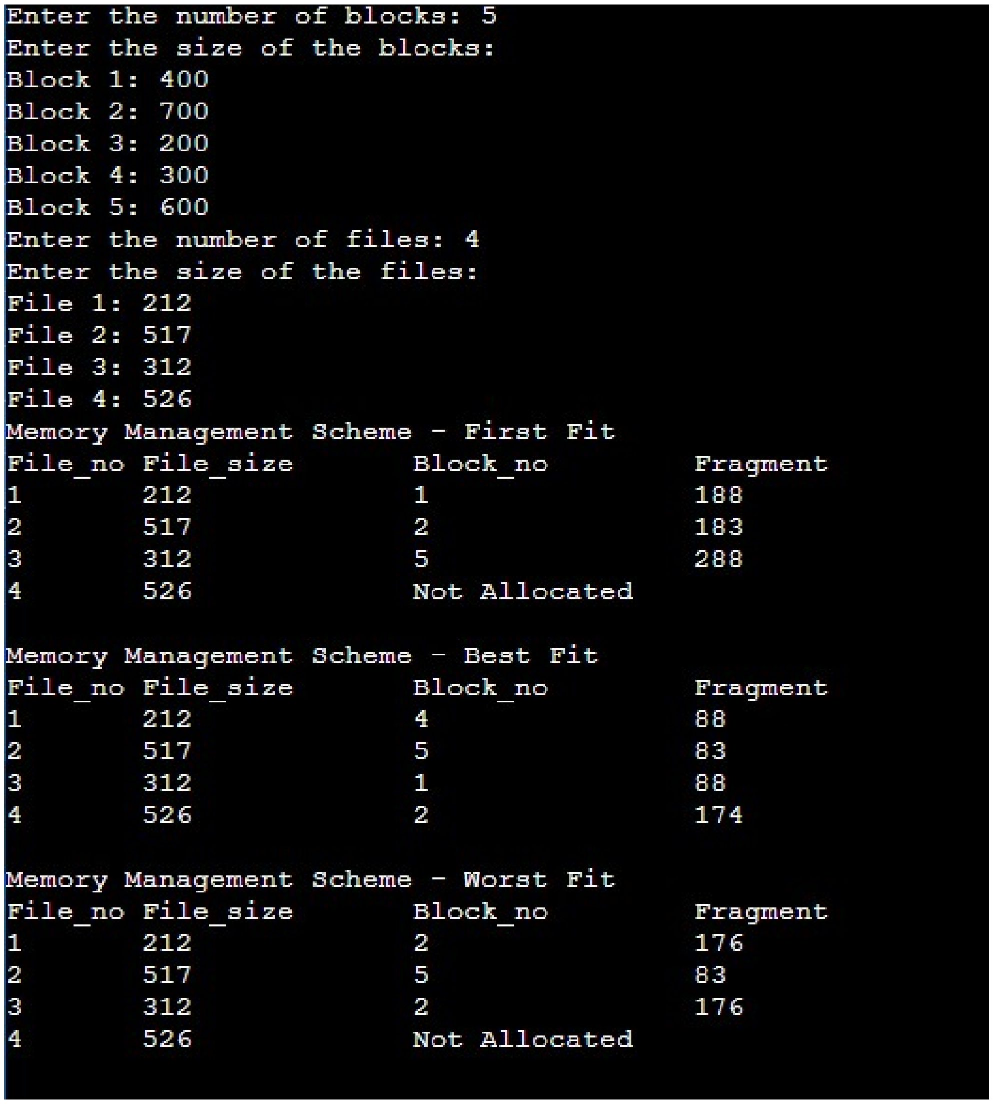


```c

        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= fileSize[i];
        }
    }

    printf("Memory Management Scheme - Best Fit\n");
    printf("File_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t", i+1, fileSize[i]);
        if (allocation[i] != -1) {
            printf("%d\t\t%d\t\t%d\n", allocation[i]+1, blockSize[allocation[i]], blockSize[allocation[i]]);
        } else {
            printf("Not Allocated\n");
        }
    }
    printf("\n");
}

void worstFit(int blockSize[], int m, int fileSize[], int n) {
    int allocation[n];

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

    for (int i = 0; i < n; i++) {
        int worstIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= fileSize[i]) {
                if (worstIdx == -1 || blockSize[worstIdx] < blockSize[j]) {
                    worstIdx = j;
                }
            }
        }

        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            blockSize[worstIdx] -= fileSize[i];
        }
    }

    printf("Memory Management Scheme - Worst Fit\n");
    printf("File_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t", i+1, fileSize[i]);
        if (allocation[i] != -1) {
            printf("%d\t\t%d\t\t%d\n", allocation[i]+1, blockSize[allocation[i]], blockSize[allocation[i]]);
```



```c
            printf("%d\t\t%d\t\t%d\n", allocation[i]+1, blockSize[allocation[i]], blockSize[allocation[i]]);
        } else {
            printf("Not Allocated\n");
        }
    }
    printf("\n");
}

int main() {
    int m, n;

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

    printf("Enter the number of files: ");
    scanf("%d", &n);
    int fileSize[n];
    printf("Enter the size of the files:\n");
    for (int i = 0; i < n; i++) {
        printf("File %d: ", i+1);
        scanf("%d", &fileSize[i]);
    }

    int blockSizeCopy[m];
    for (int i = 0; i < m; i++) {
        blockSizeCopy[i] = blockSize[i];
    }

    firstFit(blockSizeCopy, m, fileSize, n);

    for (int i = 0; i < m; i++) {
        blockSizeCopy[i] = blockSize[i];
    }

    bestFit(blockSizeCopy, m, fileSize, n);

    for (int i = 0; i < m; i++) {
        blockSizeCopy[i] = blockSize[i];
    }

    worstFit(blockSizeCopy, m, fileSize, n);
    return 0;
}
```

OUTPUT:



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

Memory Management Scheme - Best Fit
File_no File_size       Block_no        Fragment
1       212             4               88
2       517             5               83
3       312             1               88
4       526             2               174

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