PROGRAM 9
Write a C program to simulate the following contiguous memory allocation techniques
a) Worst-fit
b) Best-fit
c) First-fit

```
#include <stdio.h>

struct Block {
    int block_no;
    int block_size;
    int is_free; // 1 for free, 0 for allocated
};

struct File {
    int file_no;
    int file_size;
};

void firstFit(struct Block blocks[], int n_blocks, struct File files[], int n_files) {
    printf("Memory Management Scheme - First Fit\n");
    printf("File_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment\n");

    for (int i = 0; i < n_files; i++) {
        for (int j = 0; j < n_blocks; j++) {
            if (blocks[j].is_free && blocks[j].block_size >= files[i].file_size) {
                blocks[j].is_free = 0;
                printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", files[i].file_no, files[i].file_size, blocks[j].block_no,
blocks[j].block_size, blocks[j].block_size - files[i].file_size);
                break;
            }
        }
    }
}

void worstFit(struct Block blocks[], int n_blocks, struct File files[], int n_files) {
    printf("Memory Management Scheme - Worst Fit\n");
    printf("File_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment\n");

    for (int i = 0; i < n_files; i++) {
        int worst_fit_block = -1;
        int max_fragment = -1; // Initialize with a negative value
```

```
        for (int j = 0; j < n_blocks; j++) {
            if (blocks[j].is_free && blocks[j].block_size >= files[i].file_size) {
                int fragment = blocks[j].block_size - files[i].file_size;
                if (fragment > max_fragment) {
                    max_fragment = fragment;
                    worst_fit_block = j;
                }
            }
        }

        if (worst_fit_block!= -1) {
            blocks[worst_fit_block].is_free = 0;
            printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", files[i].file_no, files[i].file_size,
blocks[worst_fit_block].block_no, blocks[worst_fit_block].block_size, max_fragment);
        }
    }
}

void bestFit(struct Block blocks[], int n_blocks, struct File files[], int n_files) {
    printf("Memory Management Scheme - Best Fit\n");
    printf("File_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment\n");

    for (int i = 0; i < n_files; i++) {
        int best_fit_block = -1;
        int min_fragment = 10000; // Initialize with a large value
        for (int j = 0; j < n_blocks; j++) {
            if (blocks[j].is_free && blocks[j].block_size >= files[i].file_size) {
                int fragment = blocks[j].block_size - files[i].file_size;
                if (fragment < min_fragment) {
                    min_fragment = fragment;
                    best_fit_block = j;
                }
            }
        }

        if (best_fit_block!= -1) {
            blocks[best_fit_block].is_free = 0;
            printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", files[i].file_no, files[i].file_size,
blocks[best_fit_block].block_no, blocks[best_fit_block].block_size, min_fragment);
        }
    }
}

int main() {
```

```
    int n_blocks, n_files;
    printf("Enter the number of blocks: ");
    scanf("%d", &n_blocks);
    printf("Enter the number of files: ");
    scanf("%d", &n_files);

    struct Block blocks[n_blocks];
    for (int i = 0; i < n_blocks; i++) {
        blocks[i].block_no = i + 1;
        printf("Enter the size of block %d: ", i + 1);
        scanf("%d", &blocks[i].block_size);
        blocks[i].is_free = 1;
    }

    struct File files[n_files];
    for (int i = 0; i < n_files; i++) {
        files[i].file_no = i + 1;
        printf("Enter the size of file %d: ", i + 1);
        scanf("%d", &files[i].file_size);
    }

    firstFit(blocks, n_blocks, files, n_files);
    printf("\n");

    // Reset blocks for worst fit
    for (int i = 0; i < n_blocks; i++) {
        blocks[i].is_free = 1;
    }

    worstFit(blocks, n_blocks, files, n_files);
    printf("\n");

// Reset blocks for best fit
    for (int i = 0; i < n_blocks; i++) {
        blocks[i].is_free = 1;
    }

    bestFit(blocks, n_blocks, files, n_files);

    return 0;
}
```

OUTPUT:

```
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of block 1: 5
Enter the size of block 2: 2
Enter the size of block 3: 7
Enter the size of file 1: 1
Enter the size of file 2: 4
Memory Management Scheme - First Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               1               5               4
2               4               3               7               3

Memory Management Scheme - Worst Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               3               7               6
2               4               1               5               1

Memory Management Scheme - Best Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               2               2               1
2               4               1               5               1

Process returned 0 (0x0)   execution time : 17.597 s
Press any key to continue.
```