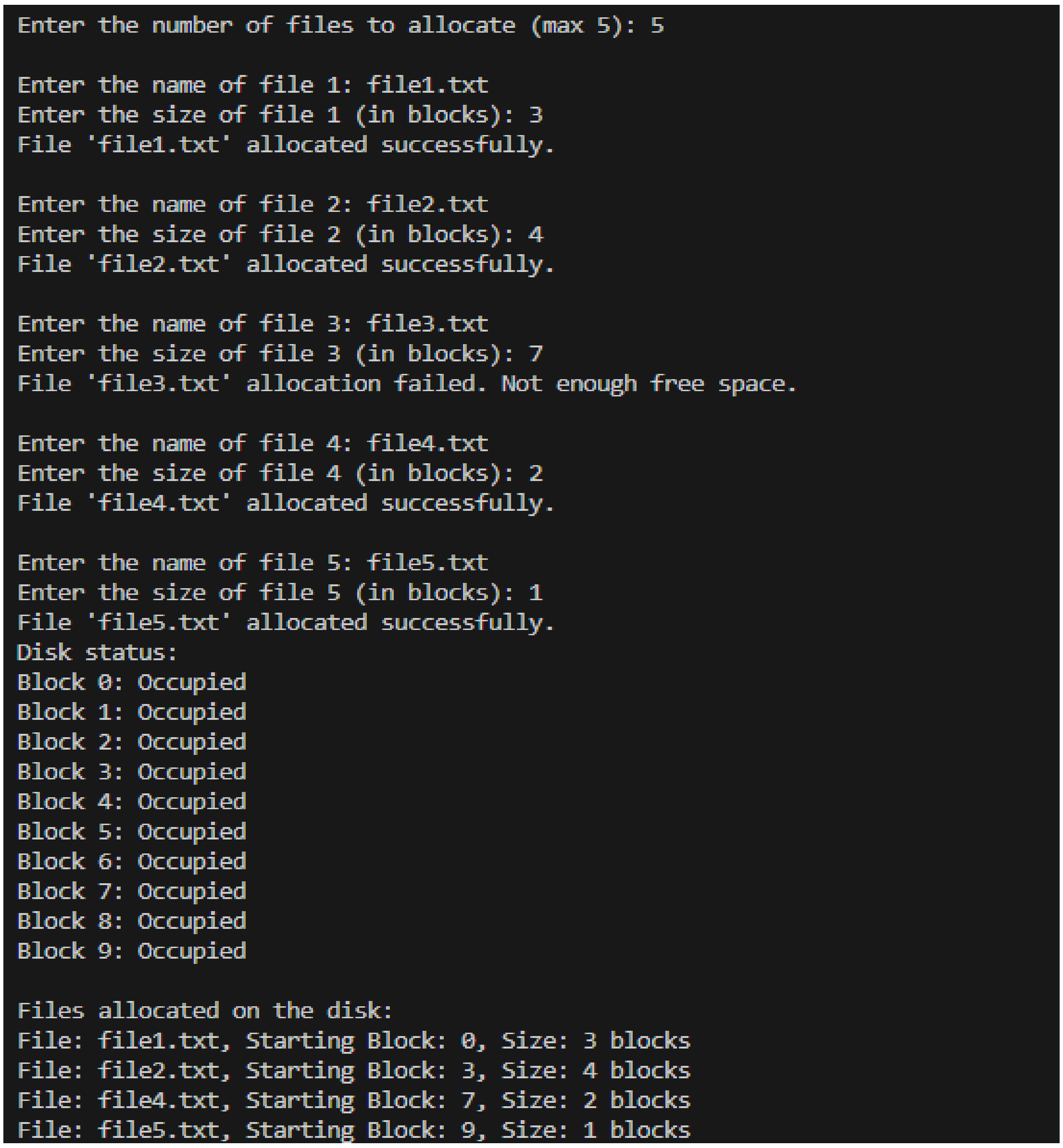
LAB 9

9) Write a C program to simulate the following file allocation strategies:

a) Sequential

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_BLOCKS 10
#define MAX_FILES 5

struct File {
    char name[20];
    int startingBlock;
    int size;
};

int disk[MAX_BLOCKS];

struct File files[MAX_FILES];
int fileCount = 0;

void initializeDisk() {
    for (int i = 0; i < MAX_BLOCKS; i++) {
        disk[i] = 0;
    }
}

int allocateFile(char *fileName, int fileSize) {
    int i, j;

    for (i = 0; i < MAX_BLOCKS - fileSize + 1; i++) {
        int canAllocate = 1;

        for (j = 0; j < fileSize; j++) {
            if (disk[i + j] != 0) {
                canAllocate = 0;
                break;
            }
        }

        if (canAllocate) {
            for (j = 0; j < fileSize; j++) {
                disk[i + j] = 1;
            }

            struct File newFile;
            strcpy(newFile.name, fileName);
            newFile.startingBlock = i;
            newFile.size = fileSize;
            files[fileCount] = newFile;
            fileCount++;
            return 1;
        }
    }
    return 0;
}

void displayDisk() {
    printf("Disk status:\n");
    for (int i = 0; i < MAX_BLOCKS; i++) {
        printf("Block %d: %s\n", i, disk[i] == 0 ? "Free" : "Occupied");
    }
}

void displayFiles() {
    printf("\nFiles allocated on the disk:\n");
    for (int i = 0; i < fileCount; i++) {
        printf("File: %s, Starting Block: %d, Size: %d blocks\n",
               files[i].name, files[i].startingBlock, files[i].size);
    }
}

void main() {
    initializeDisk();
    int numFiles;
    printf("Enter the number of files to allocate (max %d): ", MAX_FILES);
    scanf("%d", &numFiles);
    if (numFiles > MAX_FILES) {
        printf("Error: You can allocate a maximum of %d files.\n", MAX_FILES);
        return 1;
    }
    for(int i=0;i<numFiles;i++){
        char fileName[20];
        int fileSize;
        printf("\nEnter the name of file %d: ", i + 1);
        scanf("%s", fileName);
        printf("Enter the size of file %d (in blocks): ", i + 1);
        scanf("%d", &fileSize);
        if (fileSize > MAX_BLOCKS) {
            printf("Error: File size exceeds the number of available blocks on the disk.\n");
            return 1;
        }
        if (allocateFile(fileName, fileSize)) {
            printf("File '%s' allocated successfully.\n", fileName);
        } else {
            printf("File '%s' allocation failed. Not enough free space.\n", fileName);
        }
    }
    displayDisk();
    displayFiles();
}
```

OUTPUT:



```
Enter the number of files to allocate (max 5): 5

Enter the name of file 1: file1.txt
Enter the size of file 1 (in blocks): 3
File 'file1.txt' allocated successfully.

Enter the name of file 2: file2.txt
Enter the size of file 2 (in blocks): 4
File 'file2.txt' allocated successfully.

Enter the name of file 3: file3.txt
Enter the size of file 3 (in blocks): 7
File 'file3.txt' allocation failed. Not enough free space.

Enter the name of file 4: file4.txt
Enter the size of file 4 (in blocks): 2
File 'file4.txt' allocated successfully.

Enter the name of file 5: file5.txt
Enter the size of file 5 (in blocks): 1
File 'file5.txt' allocated successfully.
Disk status:
Block 0: Occupied
Block 1: Occupied
Block 2: Occupied
Block 3: Occupied
Block 4: Occupied
Block 5: Occupied
Block 6: Occupied
Block 7: Occupied
Block 8: Occupied
Block 9: Occupied

Files allocated on the disk:
File: file1.txt, Starting Block: 0, Size: 3 blocks
File: file2.txt, Starting Block: 3, Size: 4 blocks
File: file4.txt, Starting Block: 7, Size: 2 blocks
File: file5.txt, Starting Block: 9, Size: 1 blocks
```