1. Consider a file system that brings all the file pointers together into an index block. The ith entry in the index block points to the ith block of the file. Design a C program to simulate the file allocation strategy.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_BLOCKS 100

#define MAX\_FILES 10

typedef struct {

char name[20];

int indexBlock;

int blocks[20];

int blockCount;

} File;

int memory[MAX\_BLOCKS];

File files[MAX\_FILES];

int main() {

int totalBlocks, fileCount, i, j, count = 0;

printf("Enter total number of memory blocks: ");

scanf("%d", &totalBlocks);

for (i = 0; i < totalBlocks; i++)

memory[i] = 0;

printf("Enter number of files: ");

scanf("%d", &fileCount);

for (i = 0; i < fileCount; i++) {

printf("\nEnter file name: ");

scanf("%s", files[i].name);

printf("Enter number of data blocks needed: ");

scanf("%d", &files[i].blockCount);

int indexFound = 0;

for (j = 0; j < totalBlocks; j++) {

if (memory[j] == 0) {

files[i].indexBlock = j;

memory[j] = 1;

indexFound = 1;

break;

}

}

if (!indexFound) {

printf("No space for index block. Allocation failed.\n");

i--; continue;

}

int allocated = 0;

for (j = 0; j < totalBlocks && allocated < files[i].blockCount; j++) {

if (memory[j] == 0) {

memory[j] = 1;

files[i].blocks[allocated++] = j;

}

}

if (allocated < files[i].blockCount) {

printf("Not enough free blocks. Allocation failed for %s.\n", files[i].name);

memory[files[i].indexBlock] = 0; // free index block

for (j = 0; j < allocated; j++) memory[files[i].blocks[j]] = 0;

i--; continue;

}

printf("File %s allocated with index block at %d.\n", files[i].name, files[i].indexBlock);

}

printf("\nIndexed File Allocation Table:\n");

printf("File\tIndexBlock\tDataBlocks\n");

for (i = 0; i < fileCount; i++) {

printf("%s\t%d\t\t", files[i].name, files[i].indexBlock);

for (j = 0; j < files[i].blockCount; j++)

printf("%d ", files[i].blocks[j]);

printf("\n");

}

return 0;

}