1. Consider a file system where the records of the file are stored one after another both physically and logically. A record of the file can only be accessed by reading all the previous records. Design a C program to simulate the file allocation strategy.

#include <stdio.h>

#include <string.h>

#define MAX\_BLOCKS 100

#define MAX\_FILES 10

typedef struct {

char name[20];

int start;

int length;

} File;

int memory[MAX\_BLOCKS];

File files[MAX\_FILES];

int main() {

int totalBlocks, fileCount, i, j, k, found;

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 blocks required for %s: ", files[i].name);

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

found = 0;

for (j = 0; j <= totalBlocks - files[i].length; j++) {

for (k = 0; k < files[i].length; k++) {

if (memory[j + k] != 0)

break;

}

if (k == files[i].length) {

files[i].start = j;

for (k = 0; k < files[i].length; k++)

memory[j + k] = 1;

found = 1;

break;

}

}

if (!found) {

printf("Not enough contiguous space for file %s\n", files[i].name);

i--; // retry the same file

}

}

printf("\nFile Allocation Table (Sequential Allocation):\n");

printf("File\tStart\tLength\tBlocks\n");

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

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

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

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

}

printf("\n");

}

return 0;

}