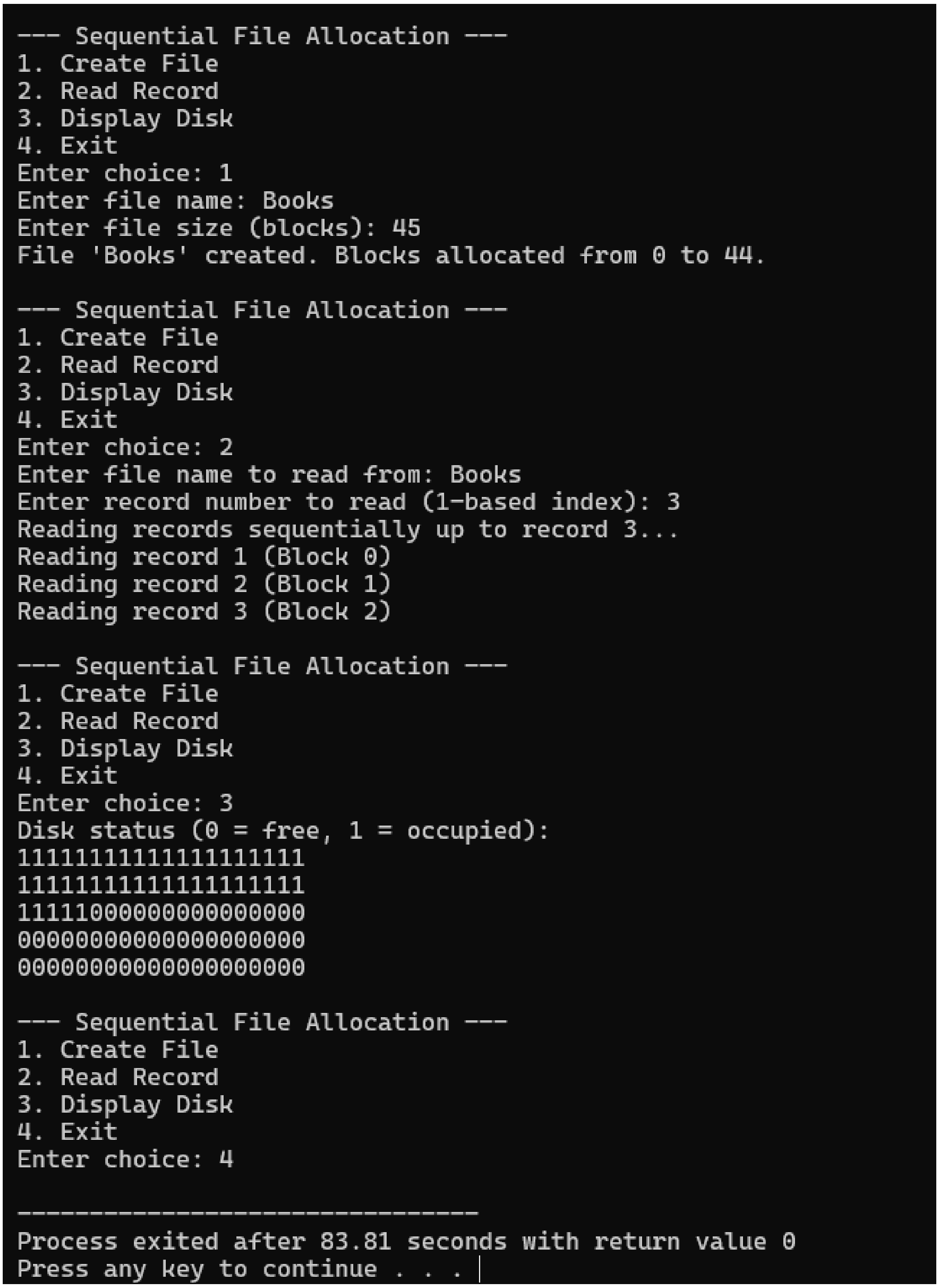
EXP 34: Consider a file system where the records of the file are stored one after another both physically and logically. A record of the file can only be accessed by reading all the previous records. Design a C program to simulate the file allocation strategy.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_BLOCKS 100

#define MAX_FILES 10


typedef struct {

    char name[20];

    int start;

    int size;

} File;

int disk[MAX_BLOCKS];  // 0 = free, 1 = occupied


File files[MAX_FILES];

int fileCount = 0;


// Allocate space sequentially

int allocateSequential(int size) {

    for (int i = 0; i <= MAX_BLOCKS - size; i++) {

        int j;

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

            if (disk[i + j] == 1)

                break;

        }
        if (j == size) { // Found sufficient contiguous space
            for (j = 0; j < size; j++)
                disk[i + j] = 1;
            return i; // return start index
        }
    }
    return -1; // Allocation failed
}
void createFile() {
    if (fileCount >= MAX_FILES) {
        printf("Maximum file limit reached!\n");
        return;
    }
    char name[20];
    int size;

    printf("Enter file name: ");
    scanf("%s", name);

    printf("Enter file size (blocks): ");
    scanf("%d", &size);

    int start = allocateSequential(size);
    if (start == -1) {
        printf("Not enough contiguous space available on disk.\n");
        return;
    }

    strcpy(files[fileCount].name, name);
    files[fileCount].start = start;

    files[fileCount].size = size;

    fileCount++;


    printf("File '%s' created. Blocks allocated from %d to %d.\n", name, start, start + size - 1);

}

void readRecord() {

    char name[20];

    int record;

    printf("Enter file name to read from: ");

    scanf("%s", name);

    printf("Enter record number to read (1-based index): ");

    scanf("%d", &record);


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

        if (strcmp(files[i].name, name) == 0) {

            if (record < 1 || record > files[i].size) {

                printf("Invalid record number.\n");

                return;

            }

            printf("Reading records sequentially up to record %d...\n", record);

            for (int r = 1; r <= record; r++) {

                printf("Reading record %d (Block %d)\n", r, files[i].start + r - 1);

            }

            return;

        }

    }

    printf("File not found.\n");

}
```

```
void displayDisk() {
    printf("Disk status (0 = free, 1 = occupied):\n");
    for (int i = 0; i < MAX_BLOCKS; i++) {
        printf("%d", disk[i]);
        if ((i + 1) % 20 == 0) printf("\n");
    }
}

int main() {
    int choice;
    while (1) {
        printf("\n--- Sequential File Allocation ---\n");
        printf("1. Create File\n");
        printf("2. Read Record\n");
        printf("3. Display Disk\n");
        printf("4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: createFile(); break;
            case 2: readRecord(); break;
            case 3: displayDisk(); break;
            case 4: exit(0);
            default: printf("Invalid choice.\n");
        }
    }
    return 0;
}
```

Sample Output



```
--- Sequential File Allocation ---
1. Create File
2. Read Record
3. Display Disk
4. Exit
Enter choice: 1
Enter file name: Books
Enter file size (blocks): 45
File 'Books' created. Blocks allocated from 0 to 44.

--- Sequential File Allocation ---
1. Create File
2. Read Record
3. Display Disk
4. Exit
Enter choice: 2
Enter file name to read from: Books
Enter record number to read (1-based index): 3
Reading records sequentially up to record 3...
Reading record 1 (Block 0)
Reading record 2 (Block 1)
Reading record 3 (Block 2)

--- Sequential File Allocation ---
1. Create File
2. Read Record
3. Display Disk
4. Exit
Enter choice: 3
Disk status (0 = free, 1 = occupied):
11111111111111111111
11111111111111111111
11111000000000000000
00000000000000000000
00000000000000000000

--- Sequential File Allocation ---
1. Create File
2. Read Record
3. Display Disk
4. Exit
Enter choice: 4

--------------------------------
Process exited after 83.81 seconds with return value 0
Press any key to continue . . .
```