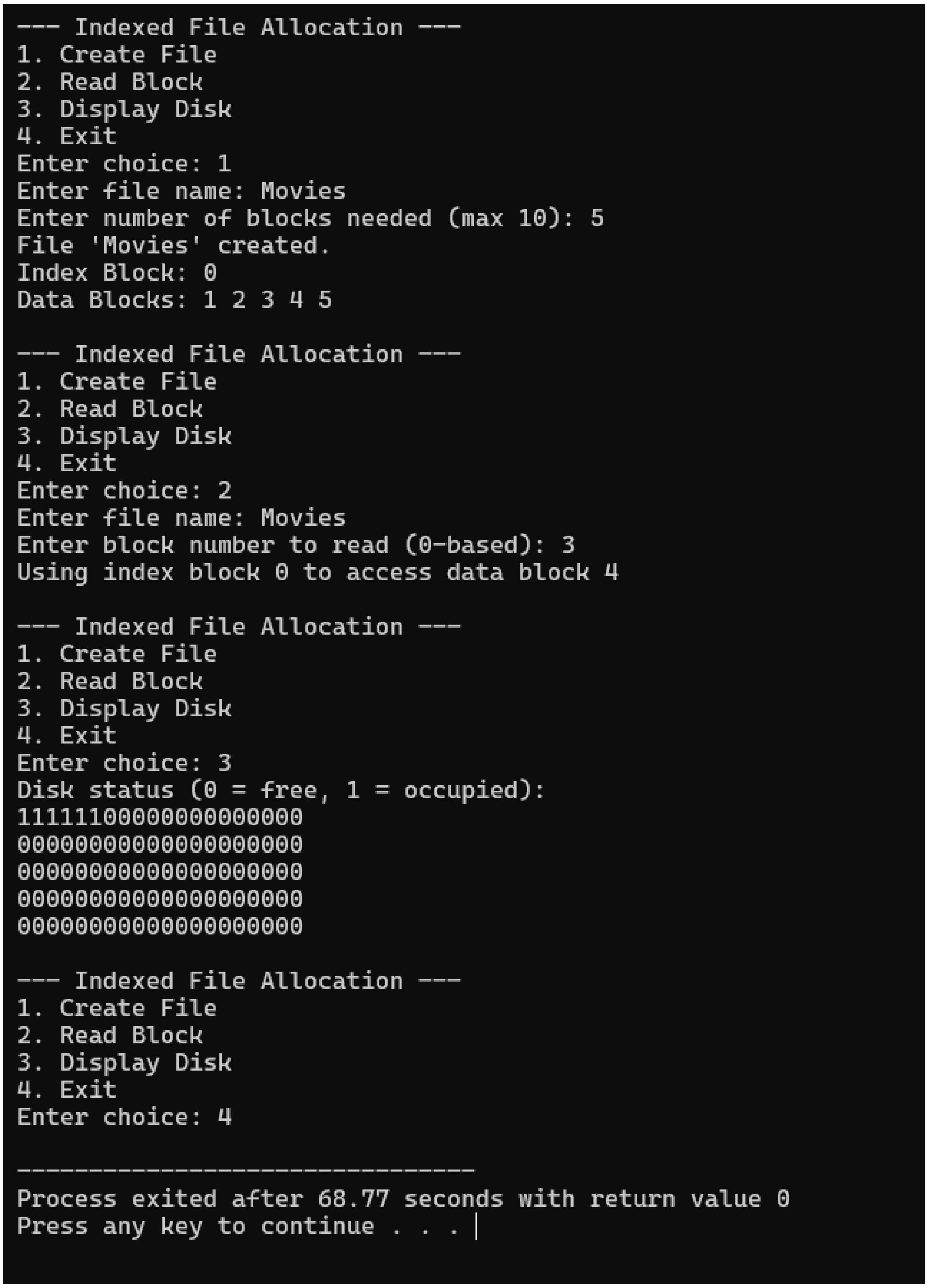
EXP 35: Consider a file system that brings all the file pointers together into an index block. The ith entry in the index block points to the ith block of the file. Design a C program to simulate the file allocation strategy.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_BLOCKS 100

#define MAX_FILES 10

#define MAX_FILE_BLOCKS 10


typedef struct {

    char name[20];

    int indexBlock;

    int blocks[MAX_FILE_BLOCKS];  // Pointers in index block

    int blockCount;

} File;


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

File files[MAX_FILES];

int fileCount = 0;


// Allocate a free block

int allocateBlock() {

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

        if (disk[i] == 0) {

            disk[i] = 1;

            return i;
        }
    }
    return -1;
}

void createFile() {
    if (fileCount >= MAX_FILES) {
        printf("Maximum file limit reached.\n");
        return;
    }

    char name[20];
    int blocks;

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

    printf("Enter number of blocks needed (max %d): ", MAX_FILE_BLOCKS);
    scanf("%d", &blocks);

    if (blocks > MAX_FILE_BLOCKS) {
        printf("Exceeded maximum blocks per file.\n");
        return;
    }

    // Allocate index block
    int indexBlock = allocateBlock();
    if (indexBlock == -1) {
        printf("No free block available for index block.\n");
        return;
    }

    int dataBlocks[MAX_FILE_BLOCKS];
    for (int i = 0; i < blocks; i++) {
        int b = allocateBlock();
        if (b == -1) {
            printf("Not enough free blocks. Rolling back allocation.\n");
            disk[indexBlock] = 0; // Free index block
            for (int j = 0; j < i; j++)
                disk[dataBlocks[j]] = 0;
            return;
        }
        dataBlocks[i] = b;
    }

    // Store file metadata
    strcpy(files[fileCount].name, name);
    files[fileCount].indexBlock = indexBlock;
    files[fileCount].blockCount = blocks;
    for (int i = 0; i < blocks; i++)
        files[fileCount].blocks[i] = dataBlocks[i];
    fileCount++;

    printf("File '%s' created.\n", name);
    printf("Index Block: %d\n", indexBlock);
    printf("Data Blocks: ");
    for (int i = 0; i < blocks; i++)
        printf("%d ", dataBlocks[i]);
    printf("\n");
}

void readBlock() {
    char name[20];
    int i;

    printf("Enter file name: ");
    scanf("%s", name);
    printf("Enter block number to read (0-based): ");
    scanf("%d", &i);

    for (int f = 0; f < fileCount; f++) {
        if (strcmp(files[f].name, name) == 0) {
            if (i < 0 || i >= files[f].blockCount) {
                printf("Invalid block number.\n");
                return;
            }
            printf("Using index block %d to access data block %d\n",
                   files[f].indexBlock, files[f].blocks[i]);
            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--- Indexed File Allocation ---\n");
        printf("1. Create File\n");
        printf("2. Read Block\n");
        printf("3. Display Disk\n");
        printf("4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

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

    return 0;
}
```

Sample Output



```
--- Indexed File Allocation ---
1. Create File
2. Read Block
3. Display Disk
4. Exit
Enter choice: 1
Enter file name: Movies
Enter number of blocks needed (max 10): 5
File 'Movies' created.
Index Block: 0
Data Blocks: 1 2 3 4 5

--- Indexed File Allocation ---
1. Create File
2. Read Block
3. Display Disk
4. Exit
Enter choice: 2
Enter file name: Movies
Enter block number to read (0-based): 3
Using index block 0 to access data block 4

--- Indexed File Allocation ---
1. Create File
2. Read Block
3. Display Disk
4. Exit
Enter choice: 3
Disk status (0 = free, 1 = occupied):
11111100000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000

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

--------------------------------
Process exited after 68.77 seconds with return value 0
Press any key to continue . . .
```