36. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block. Design a C program to simulate the file allocation strategy.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
struct File {
    int start;
    int end;
    int blocks[20];
    int size;
};
int main() {
    int disk[MAX] = {0};   // 0 = free block, 1 = allocated
    int n, i, j, totalBlocks, block;
    struct File files[10];
    int numFiles;
    printf("Enter total number of disk blocks: ");
    scanf("%d", &totalBlocks);
    printf("Enter number of files: ");
    scanf("%d", &numFiles);
    for (i = 0; i < numFiles; i++) {
        printf("\nEnter number of blocks required for File %d: ", i + 1);
        scanf("%d", &files[i].size);
        printf("Allocating blocks for File %d...\n", i + 1);
        // Allocate blocks randomly
        int allocated = 0;
        for (j = 0; j < files[i].size; j++) {
```

```
            block = rand() % totalBlocks;
            // find a free block
            while (disk[block] == 1)
             block = rand() % totalBlocks;
            disk[block] = 1;
            files[i].blocks[j] = block;
            if (j == 0)
                files[i].start = block;
            if (j == files[i].size - 1)
                files[i].end = block;
        }
        printf("File %d allocated successfully.\n", i + 1);
        printf("Start block: %d\n", files[i].start);
        printf("End block: %d\n", files[i].end);
        printf("Blocks: ");
        for (j = 0; j < files[i].size; j++) {
            if (j != files[i].size - 1)
                printf("[%d] -> ", files[i].blocks[j]);
            else
                printf("[%d] -> NULL", files[i].blocks[j]);
        }
        printf("\n");
    }
    printf("\nDisk Allocation Summary:\n");
    for (i = 0; i < numFiles; i++) {
        printf("\nFile %d:\n", i + 1);
        printf("Start Block: %d\n", files[i].start);
        printf("End Block: %d\n", files[i].end);
        printf("Blocks: ");
        for (j = 0; j < files[i].size; j++) {
```

```
            if (j != files[i].size - 1)
                printf("%d -> ", files[i].blocks[j]);
            else
                printf("%d -> NULL", files[i].blocks[j]);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT:

```
[?2004l
File 'demo.txt' created successfully.
Initial File permissions: rw-r--r--
File permissions changed to 754 (rwxr-xr--).
Updated File permissions: rwxr-xr--
```

Explanation:

Owner: rwx (read, write, execute)

Group: r-x (read, execute)

Others: r-- (read only)