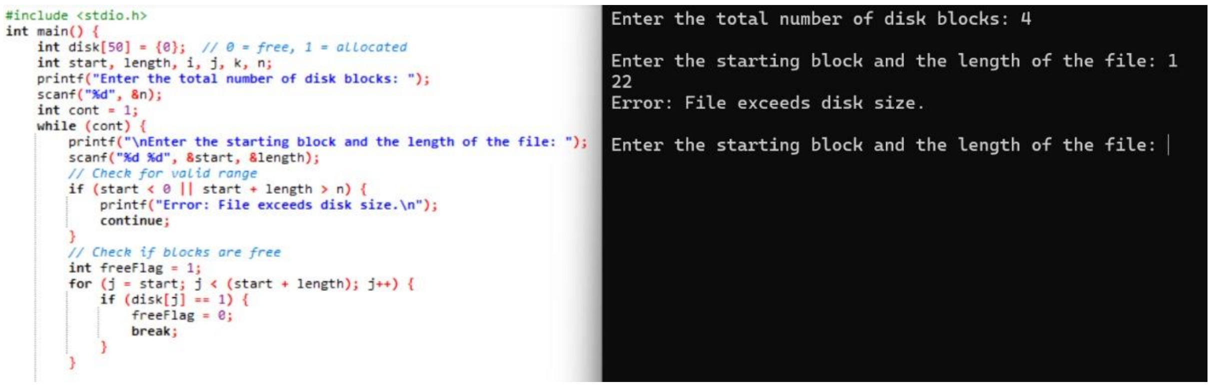
34. Consider a file system where the records of the file are stored one after another both physically and logically. A record of the file can only be accessed by reading all the previous records. Design a C program to simulate the file allocation strategy.

```
#include <stdio.h>
int main() {
   int disk[50] = {0};  // 0 = free, 1 = allocated
   int start, length, i, j, k, n;
   printf("Enter the total number of disk blocks: ");
   scanf("%d", &n);
   int cont = 1;
   while (cont) {
      printf("\nEnter the starting block and the length of the file: ");
      scanf("%d %d", &start, &length);
      // Check for valid range
      if (start < 0 || start + length > n) {
         printf("Error: File exceeds disk size.\n");
         continue;
      }
      // Check if blocks are free
      int freeFlag = 1;
      for (j = start; j < (start + length); j++) {
         if (disk[j] == 1) {
            freeFlag = 0;
            break;
         }
      }

      if (freeFlag == 1) {
            // Allocate blocks
            for (k = start; k < (start + length); k++)
                disk[k] = 1;
            printf("File allocated successfully!\n");
            printf("Blocks allocated: ");
            for (k = start; k < (start + length); k++)
                printf("%d ", k);
            printf("\n");
        } else {
            printf("Error: Blocks already allocated. File not allocated.\n");
        }
        printf("\nDo you want to enter more files? (1 = Yes, 0 = No): ");
        scanf("%d", &cont);
    }
    printf("\nFinal Disk Block Status:\n");
    for (i = 0; i < n; i++) {
        printf("Block %d: %s\n", i, disk[i] ? "Allocated" : "Free");
    }
    return 0;
}
```

OUTPUT:



```
#include <stdio.h>
int main() {
    int disk[50] = {0};  // 0 = free, 1 = allocated
    int start, length, i, j, k, n;
    printf("Enter the total number of disk blocks: ");
    scanf("%d", &n);
    int cont = 1;
    while (cont) {
        printf("\nEnter the starting block and the length of the file: ");
        scanf("%d %d", &start, &length);
        // Check for valid range
        if (start < 0 || start + length > n) {
            printf("Error: File exceeds disk size.\n");
            continue;
        }
        // Check if blocks are free
        int freeFlag = 1;
        for (j = start; j < (start + length); j++) {
            if (disk[j] == 1) {
                freeFlag = 0;
                break;
            }
        }
```

```
Enter the total number of disk blocks: 4

Enter the starting block and the length of the file: 1
22
Error: File exceeds disk size.

Enter the starting block and the length of the file: 
```