//Program 9. Develop a C program to simulate the Linked file allocation strategies.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_BLOCKS 100
struct Block {
int blockNumber;
struct Block* next;
};
void initializeDisk(struct Block* disk[], int size)
{
   for (int i = 0; i < size; i++)
   {
      disk[i] = NULL;
   }
}
void displayDisk(struct Block* disk[], int size)
{
   printf("Disk Blocks:\n");
   for (int i = 0; i < size; i++)
   {
      printf("Block %d: ", i);
      struct Block* current = disk[i];
      while (current != NULL) {
           printf("-> %d ", current->blockNumber);
            current = current->next;
      }
      printf("-> NULL\n");
    }
 }
```

```
int allocateBlock(struct Block* disk[], int size, int fileIndex, int blockNumber)
{
   struct Block* newBlock = (struct Block*)malloc(sizeof(struct Block));
   if (newBlock == NULL)
   {
      printf("Memory allocation failed.\n");
      return 0;
   }
   newBlock->blockNumber = blockNumber;
   newBlock->next = NULL;
   if(disk[fileIndex] == NULL)
   {
      disk[fileIndex] = newBlock;
   } else {
   struct Block* current = disk[fileIndex];
   while (current->next != NULL) {
   current = current->next;
   }
   current->next = newBlock;
  }
  return 1;
}
int main()
{

   struct Block* disk[MAX_BLOCKS];
   int diskSize = 10; // Change this to set the size of the disk
   initializeDisk(disk, diskSize);
// Simulating file allocation
   allocateBlock(disk, diskSize, 0, 1);
   allocateBlock(disk, diskSize, 0, 3);
   allocateBlock(disk, diskSize, 1, 2);
   allocateBlock(disk, diskSize, 2, 4);
```

```
    allocateBlock(disk, diskSize, 1, 5);
// Display the disk after allocation
displayDisk(disk, diskSize);
return 0;
}
```

OUTPUT: