D.GUNA VARDHAN
192372041

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.

AIM

To design a C program that simulates the **Linked Allocation File System**, where each file is represented as a linked list of disk blocks, and the directory contains pointers to the first and last blocks of the file. Each block contains a pointer to the next block, and the blocks may be scattered anywhere on the disk.

ALGORITHM

1. **Start**

2. Define a structure FileBlock to represent each disk block. Each block contains data and a pointer to the next block.

3. Create a structure File to represent the file, which contains pointers to the first and last blocks of the file.

4. Create functions for file operations such as adding a new block, displaying the file contents, and accessing specific blocks.

5. Implement a function to add a new block to the file, updating the directory with the first and last block pointers.

6. Implement a function to display the file contents by traversing through the linked list of blocks.

7. Implement a function to access a specific block in the file by following the linked list of blocks.

8. **Stop**

PROCEDURE

1. Include necessary libraries (stdio.h for input/output and stdlib.h for dynamic memory management).

2. Define a FileBlock structure to represent a block with data and a pointer to the next block.

3. Define a File structure that holds pointers to the first and last blocks of the file.

4. Create functions to add new blocks to the file, display file contents, and access specific blocks.

5. Initialize the file and perform operations such as adding blocks, displaying contents, and accessing specific blocks.

6. **End**

CODE:

```c
#include <stdio.h>

#include <stdlib.h>


typedef struct FileBlock {

    char data[100];

    struct FileBlock *next;

} FileBlock;


typedef struct {

    FileBlock *first;

    FileBlock *last;

} File;


File* createFile() {

    File *file = (File*)malloc(sizeof(File));

    file->first = NULL;

    file->last = NULL;

    return file;

}


void addBlock(File *file, const char *data) {

    FileBlock *newBlock = (FileBlock*)malloc(sizeof(FileBlock));
```

```c
    snprintf(newBlock->data, sizeof(newBlock->data), "%s", data);

    newBlock->next = NULL;


    if (file->first == NULL) {

        file->first = newBlock;

        file->last = newBlock;

    } else {

        file->last->next = newBlock;

        file->last = newBlock;

    }

}


void displayFile(File *file) {

    if (file->first == NULL) {

        printf("File is empty.\n");

        return;

    }


    FileBlock *current = file->first;

    while (current != NULL) {

        printf("%s\n", current->data);

        current = current->next;

    }

}


void accessBlock(File *file, int blockNum) {

    if (file->first == NULL) {

        printf("File is empty.\n");
```

```c
        return;
    }

    FileBlock *current = file->first;
    int count = 1;
    while (current != NULL && count < blockNum) {
        current = current->next;
        count++;
    }

    if (current == NULL) {
        printf("Block %d not found.\n", blockNum);
    } else {
        printf("Accessing Block %d: %s\n", blockNum, current->data);
    }
}

int main() {
    File *file = createFile();
    int choice, blockNum;
    char data[100];

    while (1) {
        printf("\nFile Allocation System (Linked Allocation)\n");
        printf("1. Add Block\n");
        printf("2. Display File\n");
        printf("3. Access a Specific Block\n");
        printf("4. Exit\n");
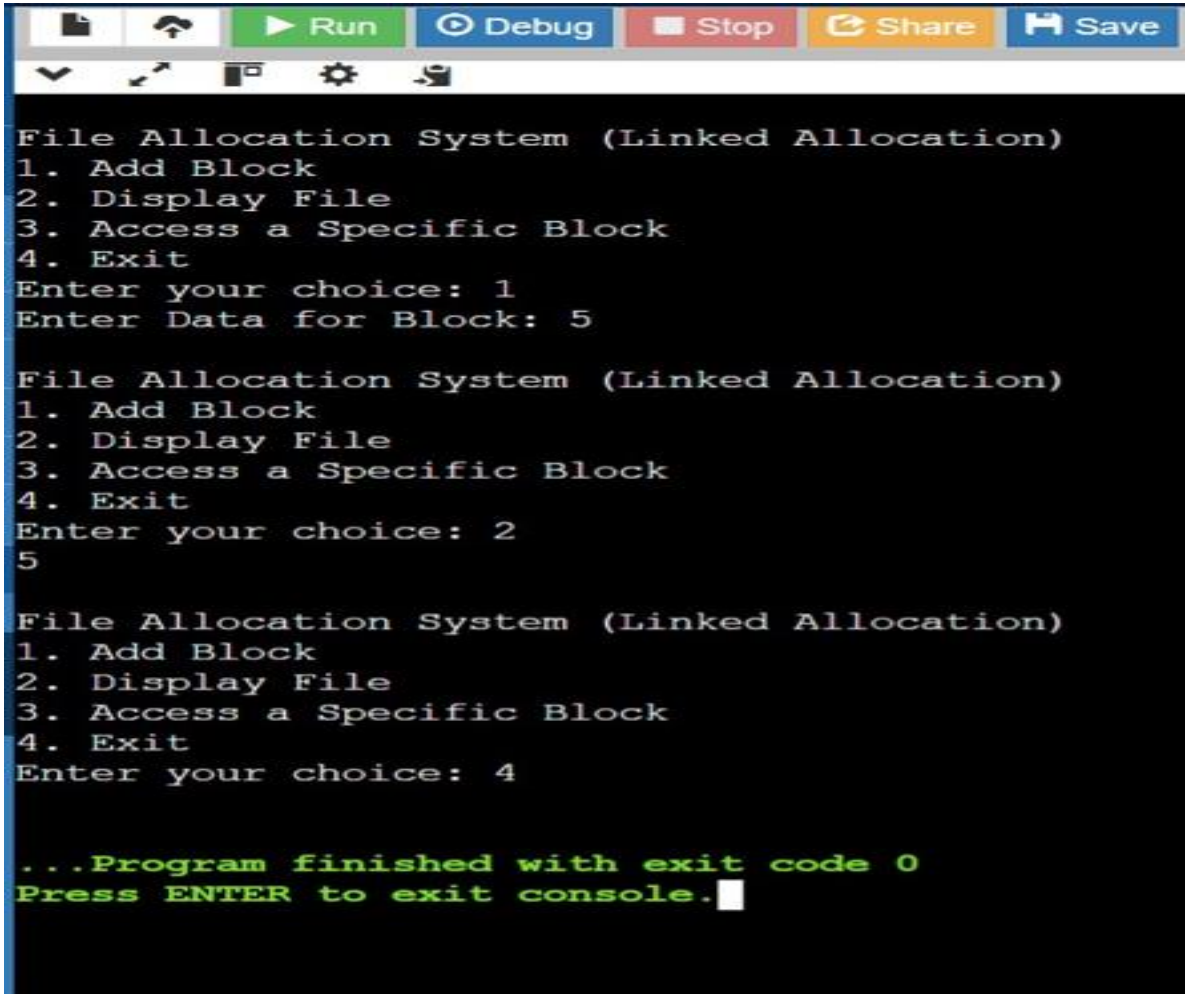```

```c
        printf("Enter your choice: ");

        scanf("%d", &choice);

        getchar();  // To consume newline character left by scanf


        switch (choice) {
          case 1:
            printf("Enter Data for Block: ");

            fgets(data, sizeof(data), stdin);

            data[strcspn(data, "\n")] = 0; // Remove newline character from input

            addBlock(file, data);

            break;
          case 2:
            displayFile(file);

            break;
          case 3:
            printf("Enter Block Number to Access: ");

            scanf("%d", &blockNum);

            accessBlock(file, blockNum);

            break;
          case 4:
            exit(0);
          default:
            printf("Invalid choice. Please try again.\n");
        }
      }


    return 0;
}
```

OUTPUT:

```
File Allocation System (Linked Allocation)
1. Add Block
2. Display File
3. Access a Specific Block
4. Exit
Enter your choice: 1
Enter Data for Block: 5

File Allocation System (Linked Allocation)
1. Add Block
2. Display File
3. Access a Specific Block
4. Exit
Enter your choice: 2
5

File Allocation System (Linked Allocation)
1. Add Block
2. Display File
3. Access a Specific Block
4. Exit
Enter your choice: 4


...Program finished with exit code 0
Press ENTER to exit console.
```