B. Siva Shirish-192324016

16. Develop a C program for implementing random access file for processing the employeedetails.

AIM:

To develop a C program that implements random access files for processing employee details.

ALGORITHM:

1. Define a structure Employee with fields such as ID, name, and salary.
2. Create a random access file where employee details will be stored.
3. Provide functionality to add, modify, delete, and display employee records.
4. Use fseek() for random access to specific records.
5. Use ftell() to determine the position in the file.
6. Implement a menu-driven program to interact with the user.

PROCEDURE:

1. Define the Employee structure with fields like ID, Name, and Salary.
2. Create a file to store employee records in binary format.
3. Implement functions for:
    - Adding a new employee to the file.
    - Modifying an existing employee's details.
    - Deleting an employee record.
    - Displaying all employee details.
4. Use fseek() to navigate to specific records by byte offset.
5. Use fwrite() and fread() to store and retrieve records from the file.
6. Implement user options to interact with the program.

CODE:

```
#include <stdio.h>
#include <string.h>

#define MAX_USERS 10
#define MAX_FILES 10
#define MAX_NAME_LENGTH 50

typedef struct {
    char fileName[MAX_NAME_LENGTH];
    int isOccupied; // 1 if file exists, 0 otherwise
} File;

typedef struct {
    char userName[MAX_NAME_LENGTH];
    File files[MAX_FILES];
    int fileCount; // Number of files in the user's directory
} UserDirectory;

UserDirectory userDirectories[MAX_USERS];
int userCount = 0;

// Function to find a user by name
int findUser(char userName[]) {
    for (int i = 0; i < userCount; i++) {
        if (strcmp(userDirectories[i].userName, userName) == 0) {
            return i;
        }
    }
    return -1;
```

```
}

// Add a new user
void addUser() {
    if (userCount >= MAX_USERS) {
        printf("Maximum user limit reached. Cannot add more users.\n");
        return;
    }

    char userName[MAX_NAME_LENGTH];
    printf("Enter the user name: ");
    scanf("%s", userName);

    if (findUser(userName) != -1) {
        printf("User '%s' already exists.\n", userName);
        return;
    }

    strcpy(userDirectories[userCount].userName, userName);
    userDirectories[userCount].fileCount = 0;
    for (int i = 0; i < MAX_FILES; i++) {
        userDirectories[userCount].files[i].isOccupied = 0;
    }
    userCount++;
    printf("User '%s' added successfully.\n", userName);
}

// Add a file to a user's directory
void addFile() {
    char userName[MAX_NAME_LENGTH], fileName[MAX_NAME_LENGTH];
```

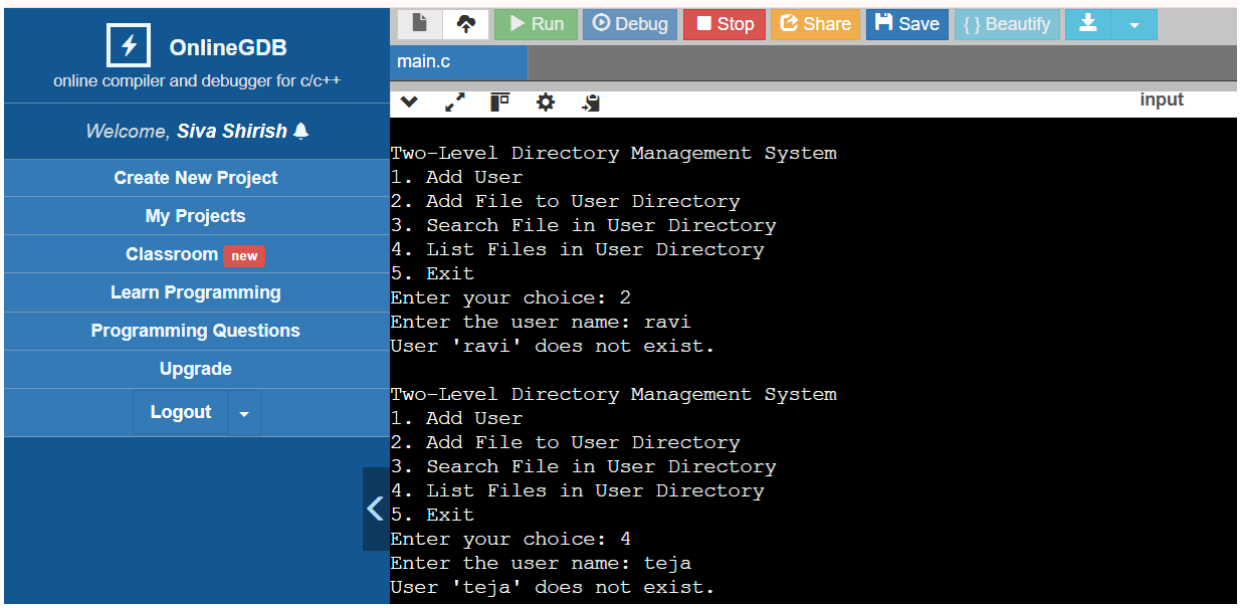
```
printf("Enter the user name: ");
scanf("%s", userName);

int userIndex = findUser(userName);
if (userIndex == -1) {
    printf("User '%s' does not exist.\n", userName);
    return;
}

if (userDirectories[userIndex].fileCount >= MAX_FILES) {
    printf("User's directory is full. Cannot add more files.\n");
    return;
}

printf("Enter the file name to add: ");
scanf("%s", fileName);

for (int i = 0; i < MAX_FILES; i++) {
    if (userDirectories[userIndex].files[i].isOccupied &&
        strcmp(userDirectories[userIndex].files[i].fileName, fileName) == 0) {
        printf("File '%s' already exists in '%s's directory.\n", fileName, userName);
        return;
    }
}

for (int i = 0; i < MAX_FILES; i++) {
    if (!userDirectories[userIndex].files[i].isOccupied) {
        strcpy(userDirectories[userIndex].files[i].fileName, fileName);
        userDirectories[userIndex].files[i].isOccupied = 1;
        userDirectories[userIndex].fileCount++;
```

```
            printf("File '%s' added successfully to '%s's directory.\n", fileName, userName);
            return;
        }
    }
}

// Search for a file in a user's directory
void searchFile() {
    char userName[MAX_NAME_LENGTH], fileName[MAX_NAME_LENGTH];
    printf("Enter the user name: ");
    scanf("%s", userName);

    int userIndex = findUser(userName);
    if (userIndex == -1) {
        printf("User '%s' does not exist.\n", userName);
        return;
    }

    printf("Enter the file name to search: ");
    scanf("%s", fileName);

    for (int i = 0; i < MAX_FILES; i++) {
        if (userDirectories[userIndex].files[i].isOccupied &&
            strcmp(userDirectories[userIndex].files[i].fileName, fileName) == 0) {
            printf("File '%s' found in '%s's directory.\n", fileName, userName);
            return;
        }
    }
    printf("File '%s' not found in '%s's directory.\n", fileName, userName);
}
```

```
// List all files in a user's directory
void listFiles() {
    char userName[MAX_NAME_LENGTH];
    printf("Enter the user name: ");
    scanf("%s", userName);

    int userIndex = findUser(userName);
    if (userIndex == -1) {
        printf("User '%s' does not exist.\n", userName);
        return;
    }

    printf("\nFiles in '%s's directory:\n", userName);
    if (userDirectories[userIndex].fileCount == 0) {
        printf("No files in the directory.\n");
    } else {
        for (int i = 0; i < MAX_FILES; i++) {
            if (userDirectories[userIndex].files[i].isOccupied) {
                printf("%s\n", userDirectories[userIndex].files[i].fileName);
            }
        }
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nTwo-Level Directory Management System\n");
```

```
        printf("1. Add User\n");
        printf("2. Add File to User Directory\n");
        printf("3. Search File in User Directory\n");
        printf("4. List Files in User Directory\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addUser();
                break;
            case 2:
                addFile();
                break;
            case 3:
                searchFile();
                break;
            case 4:
                listFiles();
                break;
            case 5:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
}
```

OUTPUT:


OnlineGDB
online compiler and debugger for c/c++
Welcome, Siva Shirish
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
main.c
input
Two-Level Directory Management System
1. Add User
2. Add File to User Directory
3. Search File in User Directory
4. List Files in User Directory
5. Exit
Enter your choice: 2
Enter the user name: ravi
User 'ravi' does not exist.
Two-Level Directory Management System
1. Add User
2. Add File to User Directory
3. Search File in User Directory
4. List Files in User Directory
5. Exit
Enter your choice: 4
Enter the user name: teja
User 'teja' does not exist.