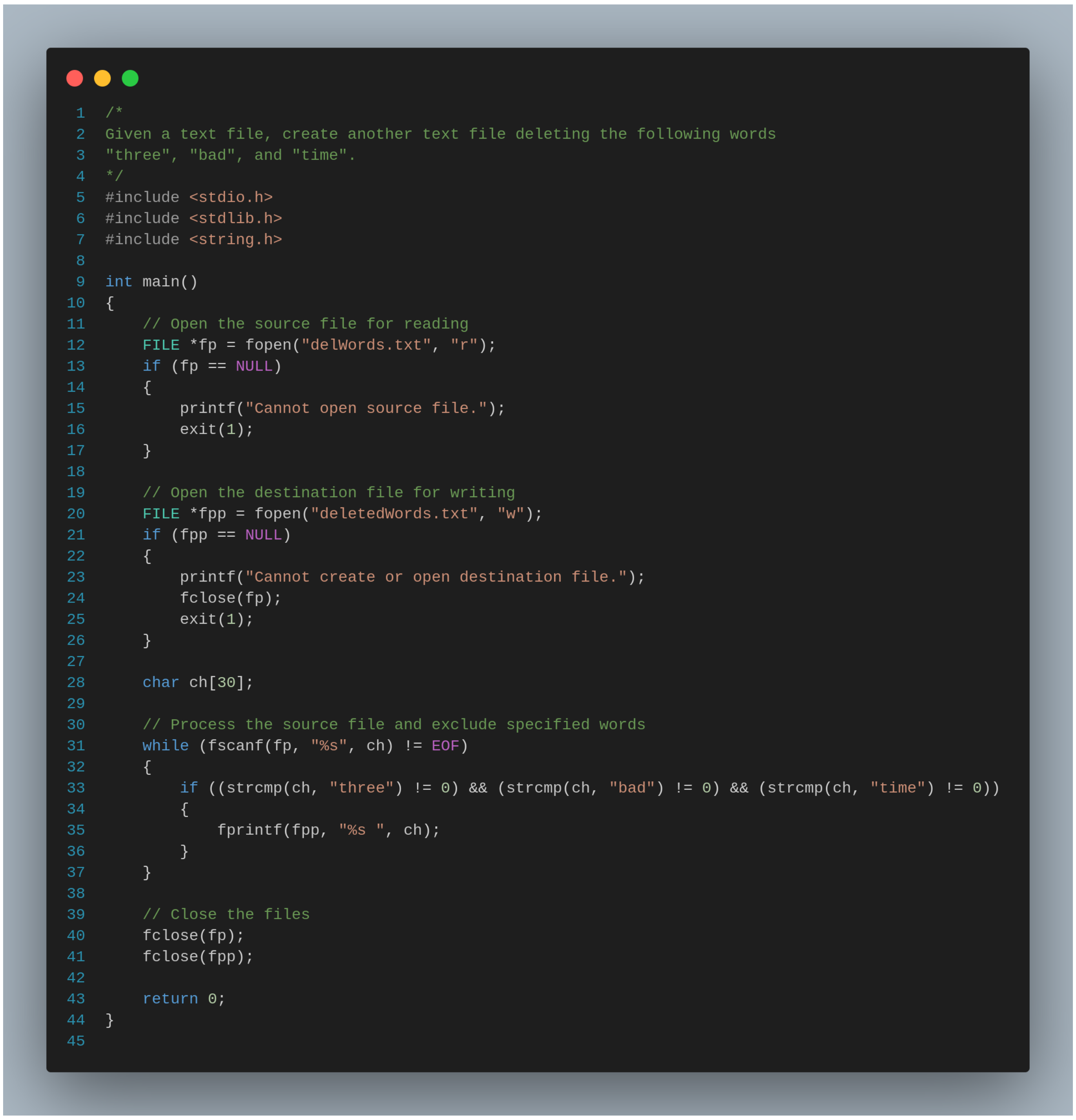
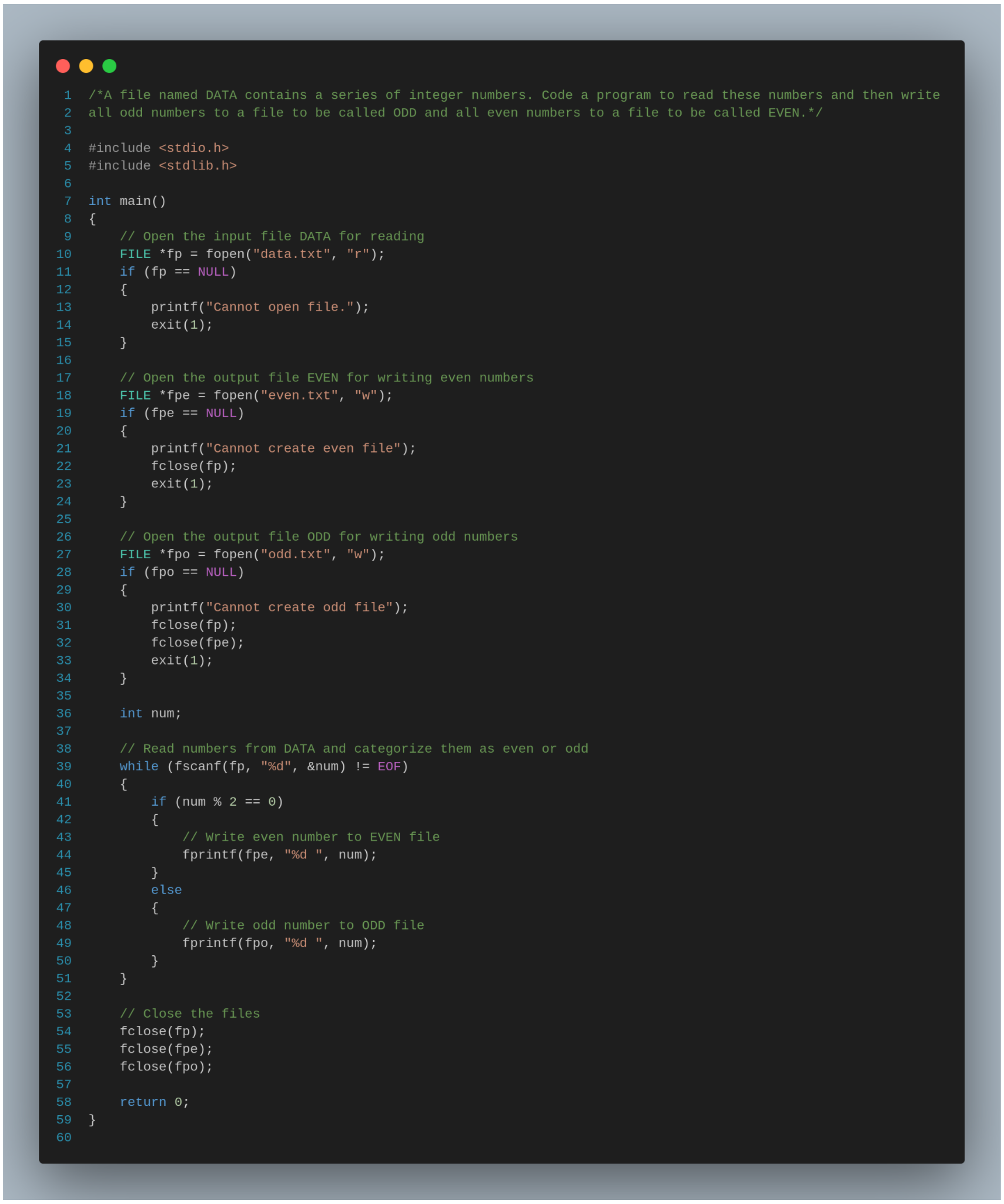
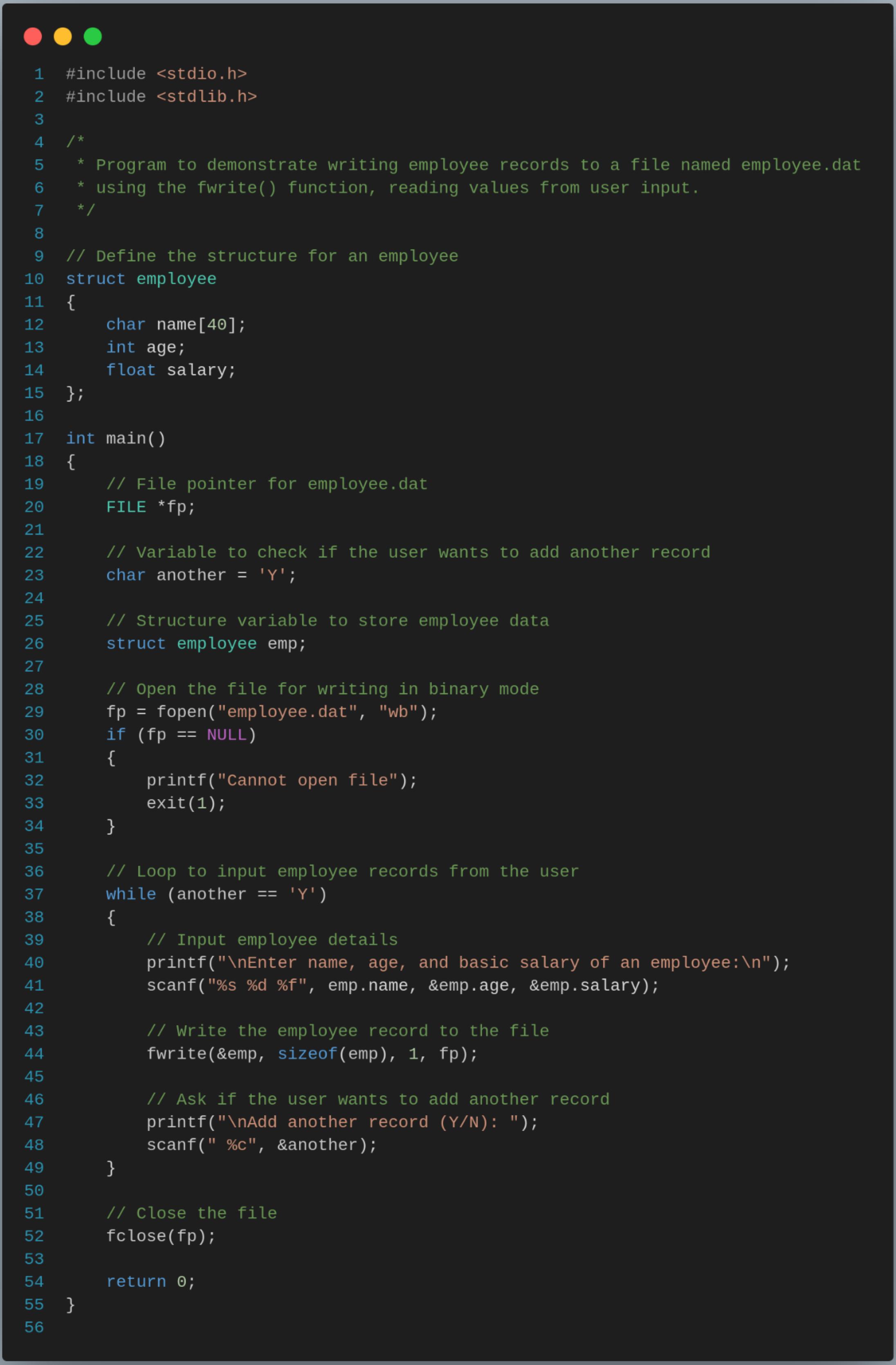


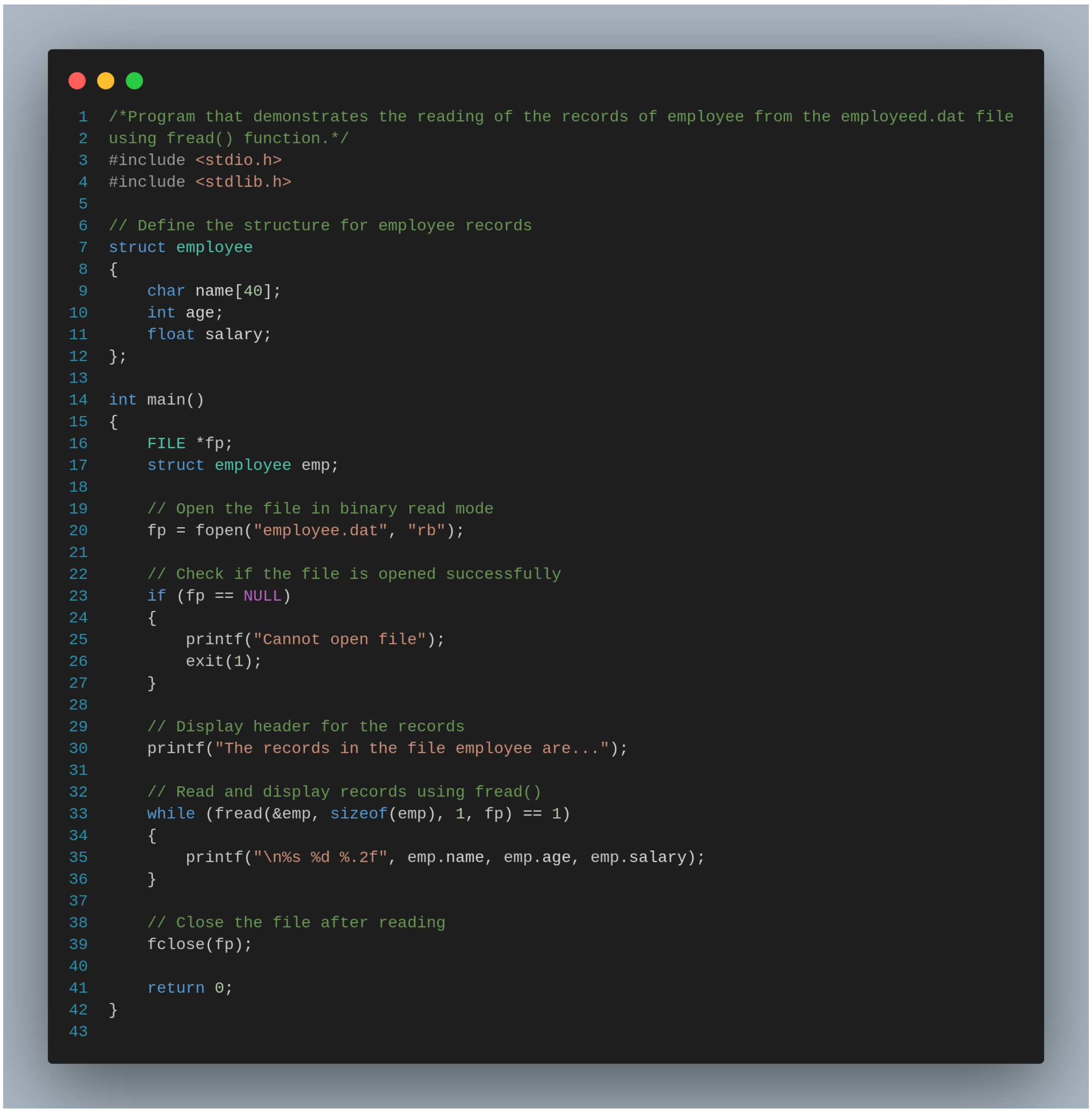
```c
/*
Given a text file, create another text file deleting the following words
"three", "bad", and "time".
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    // Open the source file for reading
    FILE *fp = fopen("delWords.txt", "r");
    if (fp == NULL)
    {
        printf("Cannot open source file.");
        exit(1);
    }

    // Open the destination file for writing
    FILE *fpp = fopen("deletedWords.txt", "w");
    if (fpp == NULL)
    {
        printf("Cannot create or open destination file.");
        fclose(fp);
        exit(1);
    }

    char ch[30];

    // Process the source file and exclude specified words
    while (fscanf(fp, "%s", ch) != EOF)
    {
        if ((strcmp(ch, "three") != 0) && (strcmp(ch, "bad") != 0) && (strcmp(ch, "time") != 0))
        {
            fprintf(fpp, "%s ", ch);
        }
    }

    // Close the files
    fclose(fp);
    fclose(fpp);

    return 0;
}
```



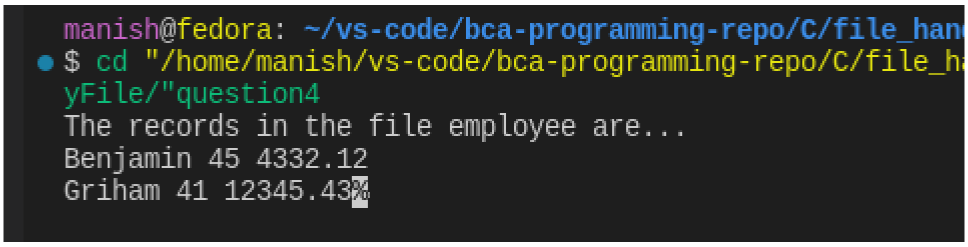
```c
/*A file named DATA contains a series of integer numbers. Code a program to read these numbers and then write
all odd numbers to a file to be called ODD and all even numbers to a file to be called EVEN.*/

#include <stdio.h>
#include <stdlib.h>

int main()
{
    // Open the input file DATA for reading
    FILE *fp = fopen("data.txt", "r");
    if (fp == NULL)
    {
        printf("Cannot open file.");
        exit(1);
    }

    // Open the output file EVEN for writing even numbers
    FILE *fpe = fopen("even.txt", "w");
    if (fpe == NULL)
    {
        printf("Cannot create even file");
        fclose(fp);
        exit(1);
    }

    // Open the output file ODD for writing odd numbers
    FILE *fpo = fopen("odd.txt", "w");
    if (fpo == NULL)
    {
        printf("Cannot create odd file");
        fclose(fp);
        fclose(fpe);
        exit(1);
    }

    int num;

    // Read numbers from DATA and categorize them as even or odd
    while (fscanf(fp, "%d", &num) != EOF)
    {
        if (num % 2 == 0)
        {
            // Write even number to EVEN file
            fprintf(fpe, "%d ", num);
        }
        else
        {
            // Write odd number to ODD file
            fprintf(fpo, "%d ", num);
        }
    }

    // Close the files
    fclose(fp);
    fclose(fpe);
    fclose(fpo);

    return 0;
}

```



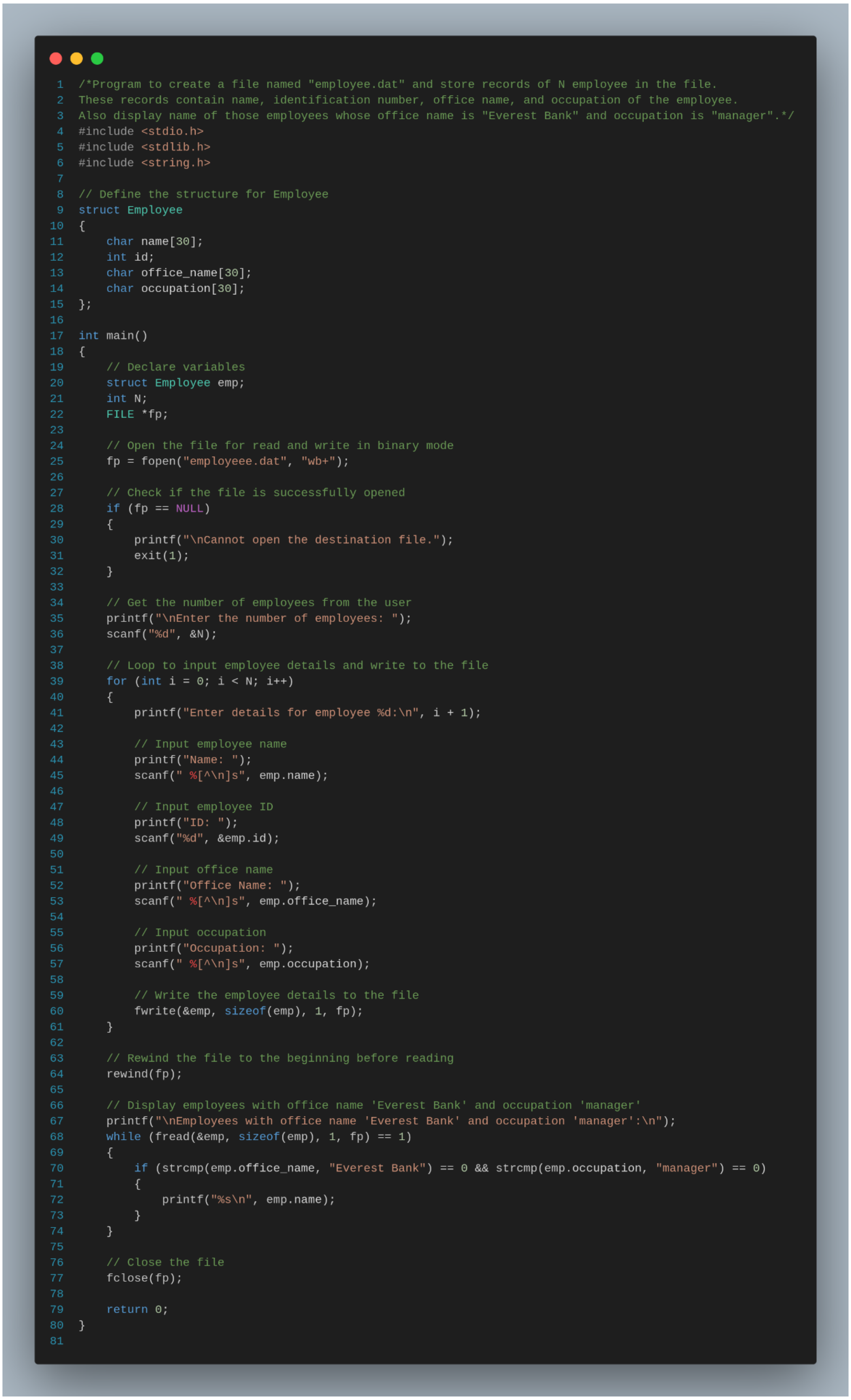
```c
#include <stdio.h>
#include <stdlib.h>

/*
 * Program to demonstrate writing employee records to a file named employee.dat
 * using the fwrite() function, reading values from user input.
 */

// Define the structure for an employee
struct employee
{
    char name[40];
    int age;
    float salary;
};

int main()
{
    // File pointer for employee.dat
    FILE *fp;

    // Variable to check if the user wants to add another record
    char another = 'Y';

    // Structure variable to store employee data
    struct employee emp;

    // Open the file for writing in binary mode
    fp = fopen("employee.dat", "wb");
    if (fp == NULL)
    {
        printf("Cannot open file");
        exit(1);
    }

    // Loop to input employee records from the user
    while (another == 'Y')
    {
        // Input employee details
        printf("\nEnter name, age, and basic salary of an employee:\n");
        scanf("%s %d %f", emp.name, &emp.age, &emp.salary);

        // Write the employee record to the file
        fwrite(&emp, sizeof(emp), 1, fp);

        // Ask if the user wants to add another record
        printf("\nAdd another record (Y/N): ");
        scanf(" %c", &another);
    }

    // Close the file
    fclose(fp);

    return 0;
}
```



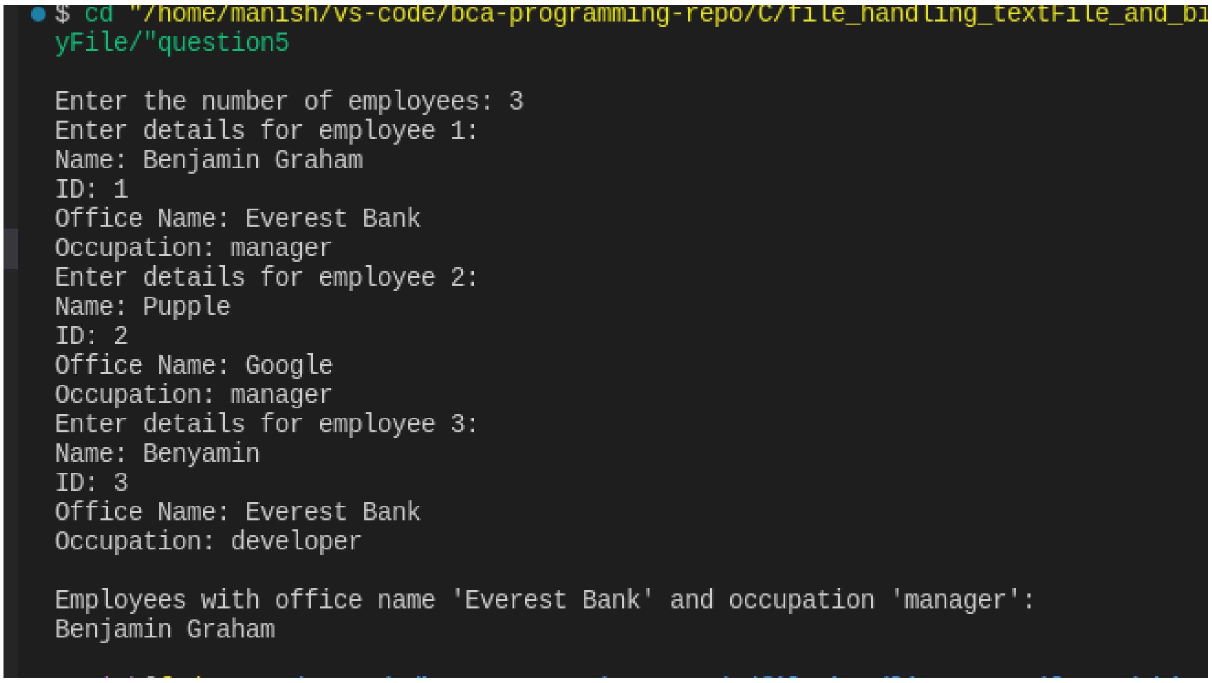
```c
/*Program that demonstrates the reading of the records of employee from the employeed.dat file
using fread() function.*/
#include <stdio.h>
#include <stdlib.h>

// Define the structure for employee records
struct employee
{
    char name[40];
    int age;
    float salary;
};

int main()
{
    FILE *fp;
    struct employee emp;

    // Open the file in binary read mode
    fp = fopen("employee.dat", "rb");

    // Check if the file is opened successfully
    if (fp == NULL)
    {
        printf("Cannot open file");
        exit(1);
    }

    // Display header for the records
    printf("The records in the file employee are...");

    // Read and display records using fread()
    while (fread(&emp, sizeof(emp), 1, fp) == 1)
    {
        printf("\n%s %d %.2f", emp.name, emp.age, emp.salary);
    }

    // Close the file after reading
    fclose(fp);

    return 0;
}

```



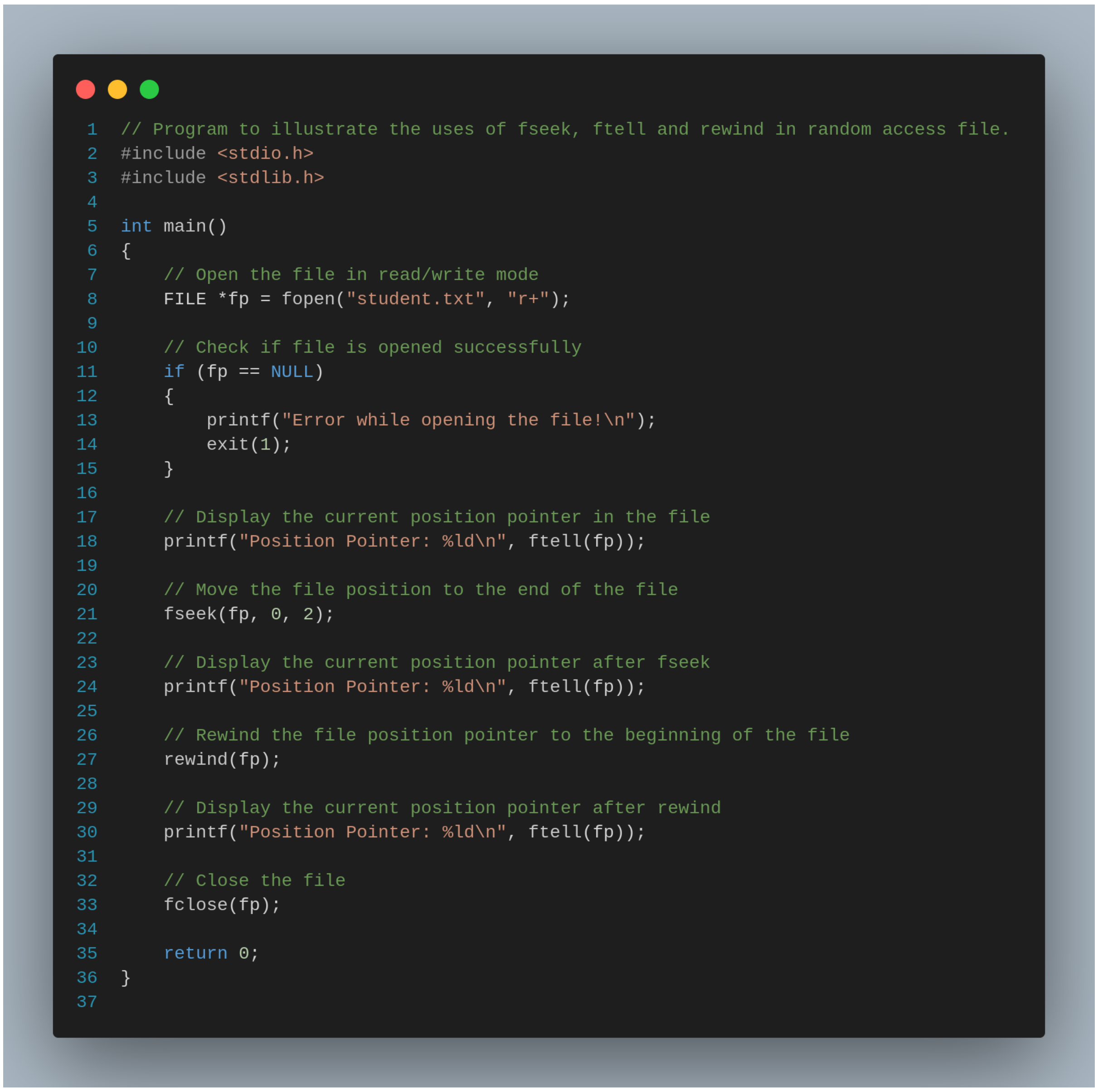
```
manish@fedora: ~/vs-code/bca-programming-repo/C/file_han
$ cd "/home/manish/vs-code/bca-programming-repo/C/file_ha
yFile/"question4
The records in the file employee are...
Benjamin 45 4332.12
Griham 41 12345.43%
```



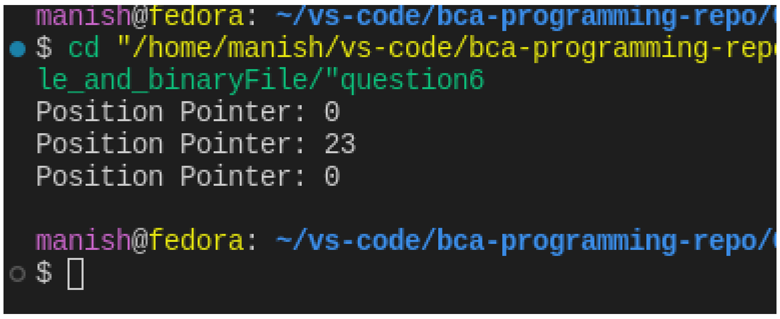
```c
/*Program to create a file named "employee.dat" and store records of N employee in the file.
These records contain name, identification number, office name, and occupation of the employee.
Also display name of those employees whose office name is "Everest Bank" and occupation is "manager".*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define the structure for Employee
struct Employee
{
    char name[30];
    int id;
    char office_name[30];
    char occupation[30];
};

int main()
{
    // Declare variables
    struct Employee emp;
    int N;
    FILE *fp;

    // Open the file for read and write in binary mode
    fp = fopen("employeee.dat", "wb+");

    // Check if the file is successfully opened
    if (fp == NULL)
    {
        printf("\nCannot open the destination file.");
        exit(1);
    }

    // Get the number of employees from the user
    printf("\nEnter the number of employees: ");
    scanf("%d", &N);

    // Loop to input employee details and write to the file
    for (int i = 0; i < N; i++)
    {
        printf("Enter details for employee %d:\n", i + 1);

        // Input employee name
        printf("Name: ");
        scanf(" %[^\n]s", emp.name);

        // Input employee ID
        printf("ID: ");
        scanf("%d", &emp.id);

        // Input office name
        printf("Office Name: ");
        scanf(" %[^\n]s", emp.office_name);

        // Input occupation
        printf("Occupation: ");
        scanf(" %[^\n]s", emp.occupation);

        // Write the employee details to the file
        fwrite(&emp, sizeof(emp), 1, fp);
    }

    // Rewind the file to the beginning before reading
    rewind(fp);

    // Display employees with office name 'Everest Bank' and occupation 'manager'
    printf("\nEmployees with office name 'Everest Bank' and occupation 'manager':\n");
    while (fread(&emp, sizeof(emp), 1, fp) == 1)
    {
        if (strcmp(emp.office_name, "Everest Bank") == 0 && strcmp(emp.occupation, "manager") == 0)
        {
            printf("%s\n", emp.name);
        }
    }

    // Close the file
    fclose(fp);

    return 0;
}
```



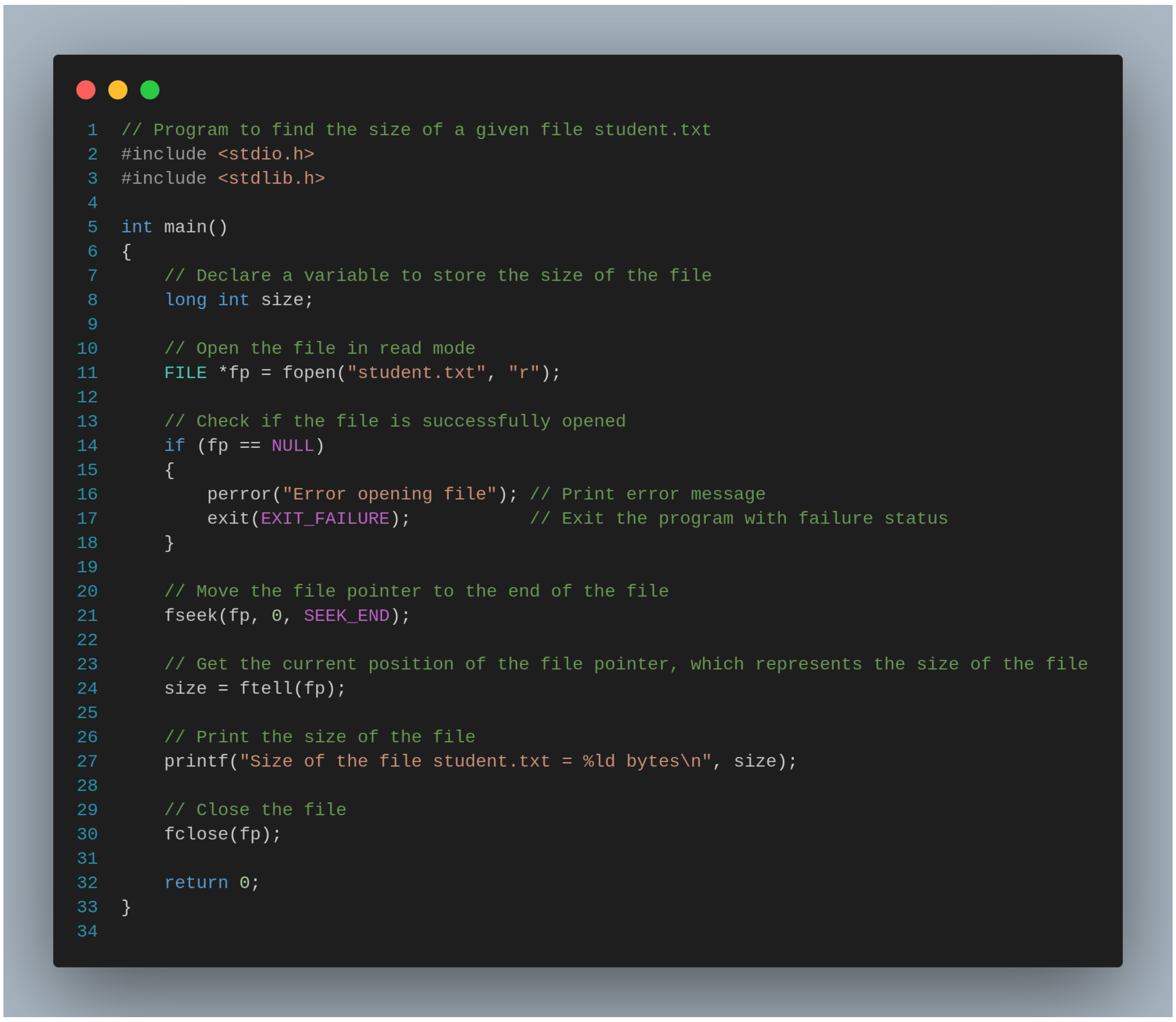
```
$ cd "/home/manish/vs-code/bca-programming-repo/C/file_handling_textFile_and_bi
yFile/"question5

Enter the number of employees: 3
Enter details for employee 1:
Name: Benjamin Graham
ID: 1
Office Name: Everest Bank
Occupation: manager
Enter details for employee 2:
Name: Pupple
ID: 2
Office Name: Google
Occupation: manager
Enter details for employee 3:
Name: Benyamin
ID: 3
Office Name: Everest Bank
Occupation: developer

Employees with office name 'Everest Bank' and occupation 'manager':
Benjamin Graham
```



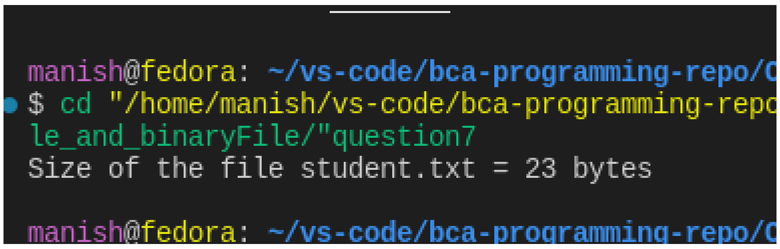
```c
// Program to illustrate the uses of fseek, ftell and rewind in random access file.
#include <stdio.h>
#include <stdlib.h>

int main()
{
    // Open the file in read/write mode
    FILE *fp = fopen("student.txt", "r+");

    // Check if file is opened successfully
    if (fp == NULL)
    {
        printf("Error while opening the file!\n");
        exit(1);
    }

    // Display the current position pointer in the file
    printf("Position Pointer: %ld\n", ftell(fp));

    // Move the file position to the end of the file
    fseek(fp, 0, 2);

    // Display the current position pointer after fseek
    printf("Position Pointer: %ld\n", ftell(fp));

    // Rewind the file position pointer to the beginning of the file
    rewind(fp);

    // Display the current position pointer after rewind
    printf("Position Pointer: %ld\n", ftell(fp));

    // Close the file
    fclose(fp);

    return 0;
}

```



```
manish@fedora: ~/vs-code/bca-programming-repo/
$ cd "/home/manish/vs-code/bca-programming-rep
le_and_binaryFile/"question6
Position Pointer: 0
Position Pointer: 23
Position Pointer: 0

manish@fedora: ~/vs-code/bca-programming-repo/
$ 
```



```c
// Program to find the size of a given file student.txt
#include <stdio.h>
#include <stdlib.h>

int main()
{
    // Declare a variable to store the size of the file
    long int size;

    // Open the file in read mode
    FILE *fp = fopen("student.txt", "r");

    // Check if the file is successfully opened
    if (fp == NULL)
    {
        perror("Error opening file"); // Print error message
        exit(EXIT_FAILURE);           // Exit the program with failure status
    }

    // Move the file pointer to the end of the file
    fseek(fp, 0, SEEK_END);

    // Get the current position of the file pointer, which represents the size of the file
    size = ftell(fp);

    // Print the size of the file
    printf("Size of the file student.txt = %ld bytes\n", size);

    // Close the file
    fclose(fp);

    return 0;
}

```



```
manish@fedora: ~/vs-code/bca-programming-repo/C
$ cd "/home/manish/vs-code/bca-programming-repo
le_and_binaryFile/"question7
Size of the file student.txt = 23 bytes

manish@fedora: ~/vs-code/bca-programming-repo/C
```