**SE LAB**

**Name:- Swapnadeep Mishra**

**Dept:- IT, Section:- A3**

**Roll no:- 002211001115**

**Assignment:- 4**

**GitHub Link:-** [**https://github.com/Deep131203/SE-Lab-ASS4**](https://github.com/Deep131203/SE-Lab-ASS4)

**1. Inventory Management System with Git**

a) Design a system to manage products for a store. Customers can make purchases, and sellers can update the list of products.

b) Use Git for version control, and maintain a purchase history of items.

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#include<time.h>**

**struct Inventory{**

**int id;**

**char productname[50];**

**int quantity;**

**float price;**

**char date[12];**

**} iv;**

**FILE \*fp;**

**void addproduct() {**

**char myDate[12];**

**time\_t t = time(NULL);**

**struct tm tm = \*localtime(&t);**

**sprintf(myDate, "%02d/%02d/%d", tm.tm\_mon + 1, tm.tm\_mday, tm.tm\_year + 1900);**

**strcpy(iv.date, myDate);**

**fp = fopen("product.txt", "ab");**

**printf("Enter product id: ");**

**scanf("%d", &iv.id);**

**printf("Enter the product name: ");**

**fflush(stdin);**

**scanf("%s", &iv.productname);**

**printf("Enter product quantity: ");**

**fflush(stdin);**

**scanf("%d", &iv.quantity);**

**printf("Enter the product price: ");**

**fflush(stdin);**

**scanf("%f", &iv.price);**

**printf("\nProduct added successfully...\n");**

**fwrite(&iv, sizeof(iv), 1, fp);**

**fclose(fp);**

**}**

**void displayproducts() {**

**system("cls");**

**printf("<=== Product List ===>\n\n");**

**printf("%-10s %-30s %-15s %-15s %s\n", "Id", "Product Name", "Quantity", "Price", "Date");**

**printf("-----------------------------------------------------------------\n");**

**fp = fopen("product.txt", "rb");**

**while (fread(&iv, sizeof(iv), 1, fp) == 1) {**

**printf("%-10d %-30s %-15d %-15f %s\n", iv.id, iv.productname, iv.quantity, iv.price, iv.date);**

**}**

**fclose(fp);**

**}**

**void updateinventory() {**

**int id, f;**

**system("cls");**

**printf("<== Update products ==>\n\n");**

**printf("Enter the product id to update: ");**

**scanf("%d", &id);**

**FILE \*ft;**

**fp = fopen("product.txt", "rb+");**

**while (fread(&iv, sizeof(iv), 1, fp) == 1) {**

**if (id == iv.id) {**

**f = 1;**

**printf("Select the operation to be performed\n");**

**printf("1. Update the product name\n");**

**printf("2. Update the quantity\n");**

**printf("3. Update the product price\n");**

**int val;**

**printf("Enter your choice: ");**

**scanf("%d", &val);**

**switch (val) {**

**case 1:**

**printf("Enter the product name: ");**

**fflush(stdin);**

**scanf("%s", &iv.productname);**

**break;**

**case 2:**

**printf("Enter product quantity: ");**

**fflush(stdin);**

**scanf("%d", &iv.quantity);**

**break;**

**case 3:**

**printf("Enter the product price: ");**

**fflush(stdin);**

**scanf("%f", &iv.price);**

**break;**

**default:**

**printf("Invalid input\n");**

**}**

**fseek(fp, -sizeof(iv), 1);**

**fwrite(&iv, sizeof(iv), 1, fp);**

**fclose(fp);**

**break;**

**}**

**}**

**if (f == 1) {**

**printf("\nProduct updated...\n");**

**} else {**

**printf("\nProduct Not Found !!");**

**}**

**}**

**void del(int id) {**

**int f = 0;**

**FILE \*ft;**

**fp = fopen("product.txt", "rb");**

**ft = fopen("temp.txt", "wb");**

**while (fread(&iv, sizeof(iv), 1, fp) == 1) {**

**if (id == iv.id) {**

**f = 1;**

**} else {**

**fwrite(&iv, sizeof(iv), 1, ft);**

**}**

**}**

**fclose(fp);**

**fclose(ft);**

**remove("product.txt");**

**rename("temp.txt", "product.txt");**

**}**

**void deleteproduct() {**

**int id, f;**

**system("cls");**

**printf("<== Delete Products ==>\n\n");**

**printf("Enter the product id to delete: ");**

**scanf("%d", &id);**

**FILE \*ft;**

**fp = fopen("product.txt", "rb");**

**while (fread(&iv, sizeof(iv), 1, fp) == 1) {**

**if (id == iv.id) {**

**f = 1;**

**fclose(fp);**

**break;**

**}**

**}**

**if (f == 1) {**

**printf("Product deleted successfully...\n");**

**del(id);**

**} else {**

**printf("\nProduct Not Found !!");**

**}**

**}**

**void administrator() {**

**int val;**

**printf("\n");**

**printf("1. Add product\n");**

**printf("2. Update inventory\n");**

**printf("3. Delete product\n");**

**printf("4. Display products\n");**

**scanf("%d", &val);**

**switch (val) {**

**case 1:**

**addproduct();**

**break;**

**case 2:**

**updateinventory();**

**break;**

**case 3:**

**deleteproduct();**

**break;**

**case 4:**

**displayproducts();**

**break;**

**default:**

**printf("Invalid input\n");**

**}**

**}**

**void buy() {**

**int id, f = 0, quant;**

**system("cls");**

**printf("<== Buy products ==>\n\n");**

**printf("Enter the product id to buy: ");**

**scanf("%d", &id);**

**printf("Enter the quantity of the product: ");**

**scanf("%d", &quant);**

**FILE \*ft;**

**float p;**

**fp = fopen("product.txt", "rb+");**

**while (fread(&iv, sizeof(iv), 1, fp) == 1) {**

**if (id == iv.id) {**

**p = iv.price;**

**if (iv.quantity - quant < 0) {**

**printf("Insufficient quantity available\n");**

**return;**

**} else if (iv.quantity - quant >= 0) {**

**f = 1;**

**iv.quantity = iv.quantity - quant;**

**fseek(fp, -sizeof(iv), 1);**

**fwrite(&iv, sizeof(iv), 1, fp);**

**fclose(fp);**

**if (iv.quantity == 0) {**

**del(iv.id);**

**}**

**break;**

**}**

**}**

**}**

**if (f == 1) {**

**printf("<===== Here is the invoice =====>\n");**

**printf("Total amount payable: %f\n", p \* quant);**

**printf("Product bought successfully...\n");**

**} else {**

**printf("Product Not Found !!\n");**

**}**

**}**

**void customer() {**

**int val;**

**printf("1. Buy product\n");**

**printf("2. View product inventory\n");**

**printf("Enter the choice: ");**

**scanf("%d", &val);**

**switch (val) {**

**case 1:**

**buy();**

**break;**

**case 2:**

**displayproducts();**

**break;**

**default:**

**printf("Invalid input\n");**

**}**

**}**

**int main() {**

**int val;**

**while (1) {**

**printf("1. Administator\n");**

**printf("2. Customer\n");**

**printf("0. Exit\n");**

**printf("-->Enter your choice: ");**

**scanf("%d", &val);**

**switch (val) {**

**case 1:**

**administrator();**

**break;**

**case 2:**

**customer();**

**break;**

**case 0:**

**exit(0);**

**default:**

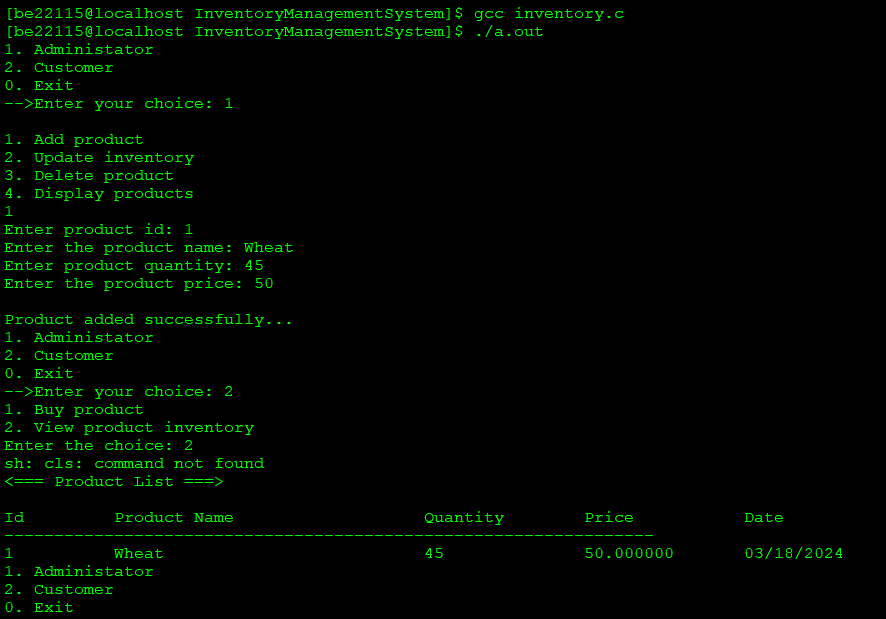
**printf("Invalid input\n");**

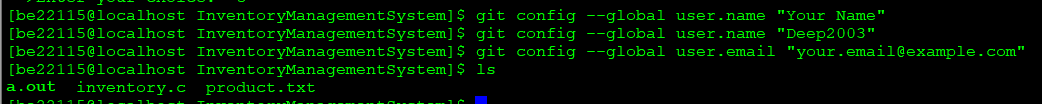
**}**

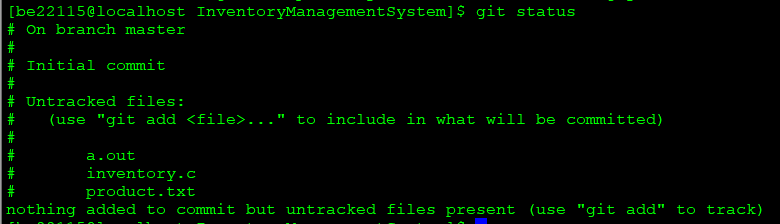
**}**

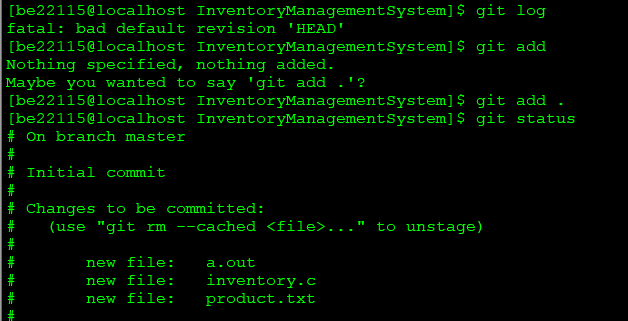
**return 0;**

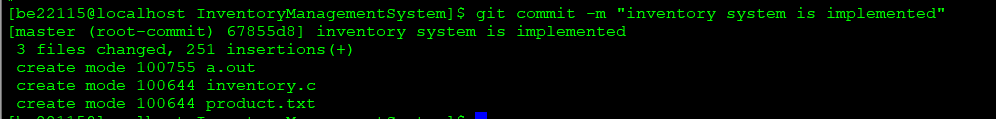
**}**











**2. Marks Management System with Git**

a) Develop a Student Marks Management System using Git.

b) In this system, a central database stores students marks for different subjects in a tabular format.

c) Subject teachers can update marks as needed before the final submission.

d) Teachers can view student names and roll numbers but only edit the marks for their subject.

e) When all teachers have completed their updates, the database is sorted by total marks and made available for students to view.

**#include <stdio.h>**

**#include <stdlib.h>**

**struct Student {**

**int rollno;**

**char name[50];**

**int marks1;**

**int marks2;**

**int marks3;**

**int marks4;**

**int marks5;**

**int totalmarks;**

**};**

**void add\_student() {**

**FILE \*fp;**

**struct Student st;**

**fp = fopen("student.txt", "ab");**

**if (fp == NULL) {**

**printf("Error opening file\n");**

**return;**

**}**

**printf("Enter name: ");**

**scanf("%s", st.name);**

**printf("Enter rollno: ");**

**scanf("%d", &st.rollno);**

**printf("Enter marks of Maths-IV: ");**

**scanf("%d", &st.marks1);**

**printf("Enter marks of Computer Networks: ");**

**scanf("%d", &st.marks2);**

**printf("Enter marks of Graph Theory: ");**

**scanf("%d", &st.marks3);**

**printf("Enter marks of OOS: ");**

**scanf("%d", &st.marks4);**

**printf("Enter marks of Software Engineering: ");**

**scanf("%d", &st.marks5);**

**st.totalmarks = st.marks1 + st.marks2 + st.marks3 + st.marks4 + st.marks5;**

**printf("Student added successfully...\n");**

**fwrite(&st, sizeof(st), 1, fp);**

**fclose(fp);**

**}**

**void update\_marks() {**

**FILE \*fp;**

**struct Student st;**

**int roll\_no, found = 0;**

**printf("\n-----Update marks-----\n");**

**printf("Enter rollno to update: ");**

**scanf("%d", &roll\_no);**

**fp = fopen("student.txt", "rb+");**

**if (fp == NULL) {**

**printf("Error opening file\n");**

**return;**

**}**

**while (fread(&st, sizeof(st), 1, fp) == 1) {**

**if (roll\_no == st.rollno) {**

**found = 1;**

**printf("Enter the new set of marks\n");**

**printf("Enter marks of Maths-IV: ");**

**scanf("%d", &st.marks1);**

**printf("Enter marks of Computer Networks: ");**

**scanf("%d", &st.marks2);**

**printf("Enter marks of Graph Theory: ");**

**scanf("%d", &st.marks3);**

**printf("Enter marks of OOS: ");**

**scanf("%d", &st.marks4);**

**printf("Enter marks of Software Engineering: ");**

**scanf("%d", &st.marks5);**

**st.totalmarks = st.marks1 + st.marks2 + st.marks3 + st.marks4 + st.marks5;**

**fseek(fp, -sizeof(st), SEEK\_CUR);**

**fwrite(&st, sizeof(st), 1, fp);**

**fclose(fp);**

**printf("\nMarks updated successfully...\n");**

**break;**

**}**

**}**

**if (!found) {**

**printf("\nStudent not found...\n");**

**}**

**}**

**void display() {**

**FILE \*fp;**

**struct Student st;**

**printf("\n---Student details----\n");**

**printf("%-10s %-30s %-15s %-20s %-20s %-20s %-20s %-20s\n", "Rollno", "Name",**

**"Maths-IV", "Computer Networks", "Graph Theory", "OOS", "Software Engineering", "Total Marks");**

**fp = fopen("student.txt", "rb");**

**if (fp == NULL) {**

**printf("Error opening file\n");**

**return;**

**}**

**while (fread(&st, sizeof(st), 1, fp) == 1) {**

**printf("%-10d %-30s %-15d %-20d %-20d %-20d %-20d %-20d\n", st.rollno, st.name,**

**st.marks1, st.marks2, st.marks3, st.marks4, st.marks5, st.totalmarks);**

**}**

**fclose(fp);**

**}**

**void teacher() {**

**int val;**

**printf("\n1. Add student\n");**

**printf("2. Update marks\n");**

**printf("3. Display marks\n");**

**printf("Enter choice: ");**

**scanf("%d", &val);**

**switch (val) {**

**case 1:**

**add\_student();**

**break;**

**case 2:**

**update\_marks();**

**break;**

**case 3:**

**display();**

**break;**

**default:**

**printf("Invalid input\n");**

**}**

**}**

**void student() {**

**display();**

**}**

**int main() {**

**int val;**

**while (1) {**

**printf("\n1. Teacher\n");**

**printf("2. Student\n");**

**printf("0. Exit\n");**

**printf("Enter choice: ");**

**scanf("%d", &val);**

**switch (val) {**

**case 1:**

**teacher();**

**break;**

**case 2:**

**student();**

**break;**

**case 0:**

**exit(0);**

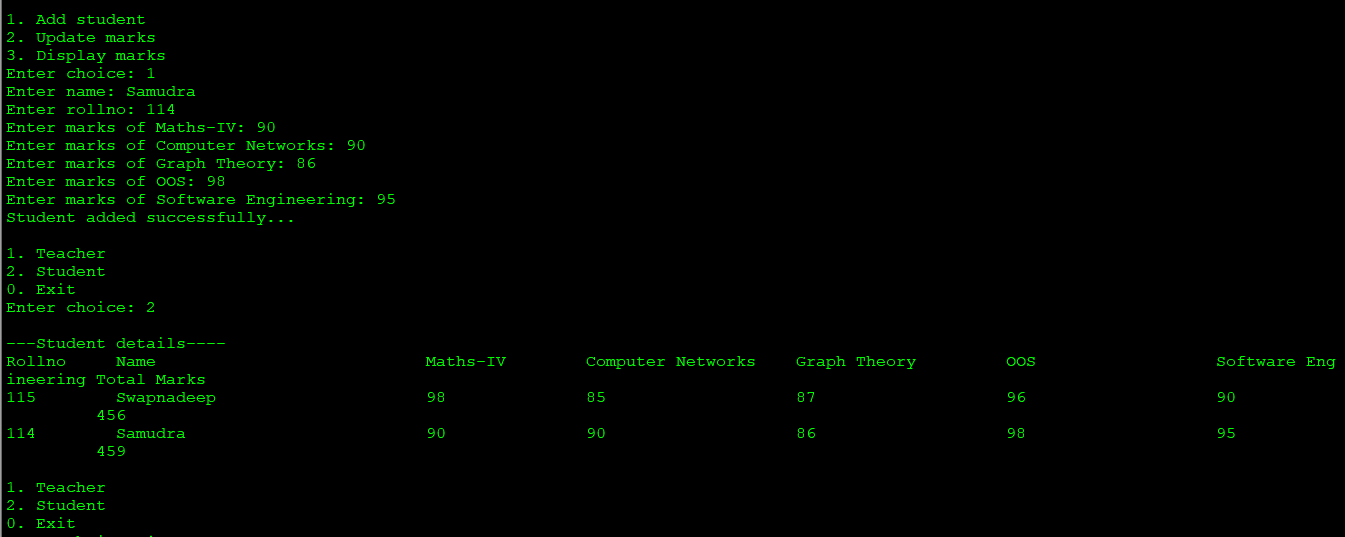
**default:**

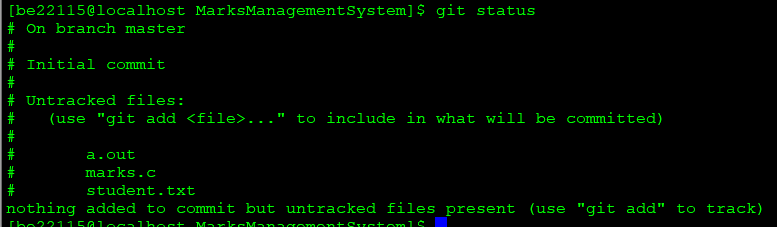
**printf("Invalid input\n");**

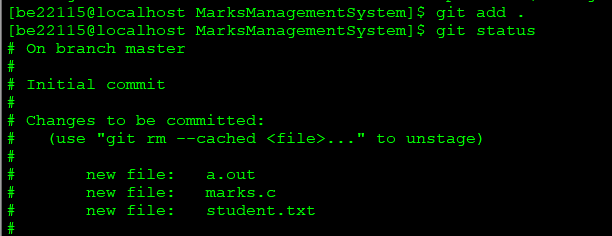
**}**

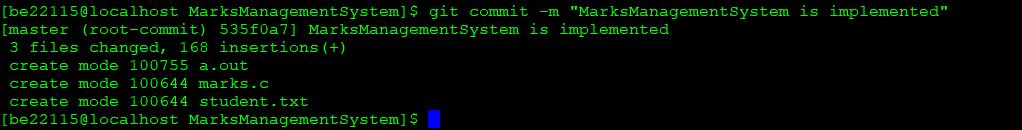
**}**

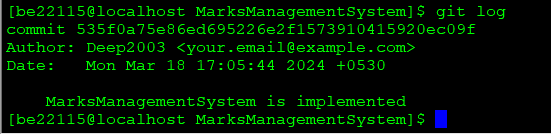
**}**











**3. Task Management CLI Tool:**

a) Develop a command-line task management tool where users can add, edit, and complete tasks.

b) Implement version control to track task changes and provide a task history.

#include <stdio.h>  
#include <stdbool.h>  
#include <string.h>  
  
#define MAX\_DESCRIPTION\_LENGTH 100  
#define MAX\_TASKS 100  
  
struct Task {  
    char description[MAX\_DESCRIPTION\_LENGTH];  
    bool completed;  
};  
  
void displayTasks(const struct Task\* tasks, size\_t numTasks) {  
    printf("Task List:\n");  
    size\_t i;  
    for (i = 0; i < numTasks; ++i) {  
        printf("[%zu] %s\n", i + 1, tasks[i].completed ? "[Completed]" : "[Pending]");  
        printf("%s\n", tasks[i].description);  
    }  
}  
  
void addTask(struct Task\* tasks, size\_t\* numTasks, const char\* description) {  
    if (\*numTasks < MAX\_TASKS) {  
        struct Task newTask;  
        strncpy(newTask.description, description, MAX\_DESCRIPTION\_LENGTH - 1);  
        newTask.description[MAX\_DESCRIPTION\_LENGTH - 1] = '\0';  
        newTask.completed = false;  
        tasks[(\*numTasks)++] = newTask;  
    } else {  
        printf("Task list is full.\n");  
    }  
}  
  
void editTask(struct Task\* tasks, size\_t numTasks, size\_t taskIndex, const char\* newDescription) {  
    if (taskIndex < numTasks) {  
        strncpy(tasks[taskIndex].description, newDescription, MAX\_DESCRIPTION\_LENGTH - 1);  
        tasks[taskIndex].description[MAX\_DESCRIPTION\_LENGTH - 1] = '\0';  
    } else {  
        printf("Invalid task index.\n");  
    }  
}  
  
void completeTask(struct Task\* tasks, size\_t numTasks, size\_t taskIndex) {  
    if (taskIndex < numTasks) {  
        tasks[taskIndex].completed = true;  
    } else {  
        printf("Invalid task index.\n");  
    }  
}  
  
void saveTasks(const struct Task\* tasks, size\_t numTasks, const char\* filename) {  
    FILE\* file = fopen(filename, "w");  
    if (file != NULL) {  
        size\_t i;  
        for (i = 0; i < numTasks; ++i) {  
            fprintf(file, "%d %s\n", tasks[i].completed, tasks[i].description);  
        }  
        fclose(file);  
    } else {  
        printf("Error: Unable to save tasks to file.\n");  
    }  
}  
  
void loadTasks(struct Task\* tasks, size\_t\* numTasks, const char\* filename) {  
    FILE\* file = fopen(filename, "r");  
    if (file != NULL) {  
        \*numTasks = 0;  
        while ((\*numTasks) < MAX\_TASKS && fscanf(file, "%d ", &tasks[\*numTasks].completed) != EOF) {  
            fgets(tasks[\*numTasks].description, MAX\_DESCRIPTION\_LENGTH, file);  
            tasks[(\*numTasks)++].description[strcspn(tasks[(\*numTasks) - 1].description, "\n")] = '\0';  
        }  
        fclose(file);  
    } else {  
        printf("Error: Unable to load tasks from file.\n");  
    }  
}  
  
int main() {  
    struct Task tasks[MAX\_TASKS];  
    size\_t numTasks = 0;  
    const char\* filename = "task\_history.txt";  
    loadTasks(tasks, &numTasks, filename);  
  
    while (1) {  
        printf("\nOptions:\n");  
        printf("1. Display Tasks\n");  
        printf("2. Add Task\n");  
        printf("3. Edit Task\n");  
        printf("4. Complete Task\n");  
        printf("5. Save and Quit\n");  
  
        int choice;  
        printf("Enter your choice: ");  
        scanf("%d", &choice);  
        getchar();  
  
        switch (choice) {  
            case 1:  
                displayTasks(tasks, numTasks);  
                break;  
  
            case 2:  
                {  
                    char description[MAX\_DESCRIPTION\_LENGTH];  
                    printf("Enter task description: ");  
                    fgets(description, MAX\_DESCRIPTION\_LENGTH, stdin);  
                    description[strcspn(description, "\n")] = '\0';  
                    addTask(tasks, &numTasks, description);  
                    break;  
                }  
  
            case 3:  
                {  
                    size\_t taskIndex;  
                    char newDescription[MAX\_DESCRIPTION\_LENGTH];  
                    printf("Enter task index to edit: ");  
                    scanf("%zu", &taskIndex);  
                    getchar();  
                    printf("Enter new task description: ");  
                    fgets(newDescription, MAX\_DESCRIPTION\_LENGTH, stdin);  
                    newDescription[strcspn(newDescription, "\n")] = '\0';  
                    editTask(tasks, numTasks, taskIndex - 1, newDescription);  
                    break;  
                }  
  
            case 4:  
                {  
                    size\_t taskIndex;  
                    printf("Enter task index to mark as completed: ");  
                    scanf("%zu", &taskIndex);  
                    getchar();  
                    completeTask(tasks, numTasks, taskIndex - 1);  
                    break;  
                }  
  
            case 5:  
                saveTasks(tasks, numTasks, filename);  
                return 0;  
  
            default:  
                printf("Invalid choice. Please try again.\n");  
        }  
    }  
    return 0;  
}

