ASSIGNMENT 4

NAME : HRITHVIK KONDALKAR
ROLL NO : 002211001088

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.

```c
#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];
};

FILE *fp;

void add_product();
void display_products();
void update_inventory();
void delete_product();
void buy_product();
void administrator();
void customer();

int main() {
    int choice;
    while(1) {
        printf("\n=== Inventory Management System ===\n");
        printf("1. Administrator\n");
        printf("2. Customer\n");
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                administrator();
                break;
            case 2:
```

```
                customer();
                break;
            case 0:
                exit(0);
            default:
                printf("Invalid input. Please try again.\n");
        }
    }
    return 0;
}

void add_product() {
    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);

    struct Inventory iv;
    strcpy(iv.date, myDate);

    fp = fopen("product.txt", "ab");
    if (fp == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    printf("Enter product id: ");
    scanf("%d", &iv.id);
    printf("Enter the product name: ");
    scanf("%s", iv.productname);
    printf("Enter product quantity: ");
    scanf("%d", &iv.quantity);
    printf("Enter the product price: ");
    scanf("%f", &iv.price);

    fwrite(&iv, sizeof(struct Inventory), 1, fp);
    fclose(fp);

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

void display_products() {
    system("cls");
    printf("<=== Product List ===>\n\n");
    printf("%-10s %-30s %-30s %-20s %s\n", "Id", "Product Name", "Quantity",
"Price", "Date");
    printf("------------------------------------------------------------------
--------------------\n");
```

```
    struct Inventory iv;
    fp = fopen("product.txt", "rb");
    if (fp == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    while(fread(&iv, sizeof(struct Inventory), 1, fp) == 1) {
        printf("%-10d %-30s %-30d %-20f %s\n", iv.id, iv.productname,
iv.quantity, iv.price, iv.date);
    }

    fclose(fp);
}

void update_inventory() {
    int id, found = 0;
    printf("<== Update products ==>\n\n");
    printf("Enter the product id to update: ");
    scanf("%d", &id);

    struct Inventory iv;
    fp = fopen("product.txt", "rb+");
    if (fp == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    while (fread(&iv, sizeof(struct Inventory), 1, fp) == 1) {
        if (id == iv.id) {
            found = 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 choice;
            printf("Enter your choice: ");
            scanf("%d", &choice);

            switch(choice) {
                case 1:
                    printf("Enter the product name: ");
                    scanf("%s", iv.productname);
                    break;
                case 2:
                    printf("Enter product quantity: ");
                    scanf("%d", &iv.quantity);
```

```
                    break;
                case 3:
                    printf("Enter the product price: ");
                    scanf("%f", &iv.price);
                    break;
                default:
                    printf("Invalid input.\n");
                    break;
            }

            fseek(fp, -sizeof(struct Inventory), SEEK_CUR);
            fwrite(&iv, sizeof(struct Inventory), 1, fp);
            fclose(fp);
            break;
        }
    }

    if(found) {
        printf("\nProduct updated successfully.\n");
    } else {
        printf("\nProduct not found.\n");
    }
}

void delete_product() {
    int id, found = 0;
    printf("<== Delete Products ==>\n\n");
    printf("Enter the product id to delete: ");
    scanf("%d", &id);

    struct Inventory iv;
    FILE *ft = fopen("temp.txt", "wb");
    if (ft == NULL) {
        printf("Error creating temporary file.\n");
        exit(1);
    }

    fp = fopen("product.txt", "rb");
    if (fp == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    while (fread(&iv, sizeof(struct Inventory), 1, fp) == 1) {
        if (id == iv.id) {
            found = 1;
        } else {
            fwrite(&iv, sizeof(struct Inventory), 1, ft);
```

```
        }
    }

    fclose(fp);
    fclose(ft);

    if(found) {
        remove("product.txt");
        rename("temp.txt", "product.txt");
        printf("Product deleted successfully.\n");
    } else {
        printf("Product not found.\n");
    }
}

void buy_product() {
    int id, found = 0, quant;
    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);

    struct Inventory iv;
    fp = fopen("product.txt", "rb+");
    if (fp == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    while (fread(&iv, sizeof(struct Inventory), 1, fp) == 1) {
        if (id == iv.id) {
            found = 1;
            if (iv.quantity - quant < 0) {
                printf("Insufficient quantity available.\n");
                return;
            } else {
                iv.quantity -= quant;
                fseek(fp, -sizeof(struct Inventory), SEEK_CUR);
                fwrite(&iv, sizeof(struct Inventory), 1, fp);
                fclose(fp);
                if (iv.quantity == 0) {
                    delete_product(id);
                }
                break;
            }
        }
    }
```

```
    if(found) {
        printf("\nProduct bought successfully.\n");
    } else {
        printf("\nProduct not found.\n");
    }
}

void administrator() {
    int choice;
    printf("\nAdministrator Menu\n");
    printf("1. Add Product\n");
    printf("2. Update Inventory\n");
    printf("3. Delete Product\n");
    printf("4. Display Products\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            add_product();
            break;
        case 2:
            update_inventory();
            break;
        case 3:
            delete_product();
            break;
        case 4:
            display_products();
            break;
        default:
            printf("Invalid input. Please try again.\n");
    }
}

void customer() {
    int choice;
    printf("\nCustomer Menu\n");
    printf("1. Buy Product\n");
    printf("2. View Product Inventory\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            buy_product();
            break;
```

```
        case 2:
            display_products();
            break;
        default:
            printf("Invalid input. Please try again.\n");
    }
}
```

```
[be2288@localhost 1]$ git init
Initialized empty Git repository in /home/usr/student/ug/yr22/be2288/secondyear/se/gitassmt/1/.git/
```

```
[be2288@localhost 1]$ vim 1.c
[be2288@localhost 1]$ [be2288@localhost 1]$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       1.c
nothing added to commit but untracked files present (use "git add" to track)
```

```
[be2288@localhost 1]$ git add 1.c
[be2288@localhost 1]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   1.c
#
```

```
[be2288@localhost 1]$ git config --global user.email "kondalkarhrithvik@gmail.com"
[be2288@localhost 1]$ git config --global user.name "hrithvik kondalkar"
[be2288@localhost 1]$ git commit -m "development complete"
[master (root-commit) c7a02e9] development complete
 1 file changed, 283 insertions(+)
 create mode 100644 1.c
```

```
<=== Product List ===>

Id          Product Name                Quantity                    Price               Date
-------------------------------------------------------------------------------------
1           odomos                      23                          65.000000           04/06/2024
2           pencil                      90                          1.000000            04/06/2024
3           bottle                      15                          300.000000          04/06/2024
4           bag                         21                          200.000000          04/06/2024
```

```
=== Inventory Management System ===
1. Administrator
2. Customer
0. Exit
```

```
Administrator Menu
1. Add Product
2. Update Inventory
3. Delete Product
4. Display Products
```

```
Customer Menu
1. Buy Product
2. View Product Inventory
Enter your choice:
```

```
[be2288@localhost 1]$ git add product.txt
[be2288@localhost 1]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   product.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       a.out
[be2288@localhost 1]$ git commit -m "database save"
[master 5fd2de4] database save
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 product.txt
```

```
[be2288@localhost 1]$ git log
commit 5fd2de4cc0fe58427ac42759123cccb241d61825
Author: hrithvik kondalkar <kondalkarhrithvik@gmail.com>
Date:   Sat Apr 6 19:54:12 2024 +0530

    database save

commit c7a02e93a23726fe75fa66978f4909ee495738d1
Author: hrithvik kondalkar <kondalkarhrithvik@gmail.com>
Date:   Sat Apr 6 19:43:05 2024 +0530

    development complete
```

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.

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

struct Student {
    int rollno;
    char name[50];
    int marks1;
    int marks2;
    int marks3;
    int marks4;
    int marks5;
    int totalmarks;
};

FILE *file_pointer;
int student_count = 0;

void add_student() {
    struct Student st;
    file_pointer = fopen("student.txt", "ab");
    if (file_pointer == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    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");
    student_count++;
    fwrite(&st, sizeof(struct Student), 1, file_pointer);
    fclose(file_pointer);
}

void update_marks() {
    int roll_no, found = 0;
    struct Student st;
    printf("\n-----Update marks-----\n");
    printf("Enter rollno to update: ");
    scanf("%d", &roll_no);
    file_pointer = fopen("student.txt", "rb+");
    if (file_pointer == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    while (fread(&st, sizeof(struct Student), 1, file_pointer) == 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(file_pointer, -sizeof(st), SEEK_CUR);
            fwrite(&st, sizeof(st), 1, file_pointer);
            fclose(file_pointer);
            break;
        }
    }
```

```
    if (found) {
        printf("Marks updated successfully...\n");
    } else {
        printf("Student not found...\n");
    }
}

void display_details() {
    system("cls");
    printf("\n---Student details----\n");
    printf("%-10s %-15s %-5s %-5s %-5s %-5s %-5s %-20s\n", "Rollno", "Name",
           "Math", "CN", "GT", "OOS", "SE", "Total Marks");
    file_pointer = fopen("student.txt", "rb");
    if (file_pointer == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    struct Student st;
    while (fread(&st, sizeof(struct Student), 1, file_pointer) == 1) {
        printf("%-10d %-15s %-5d %-5d %-5d %-5d %-5d %-5d\n", st.rollno,
st.name,
               st.marks1, st.marks2, st.marks3, st.marks4, st.marks5,
st.totalmarks);
    }

    fclose(file_pointer);
}

void teacher_menu() {
    int choice;
    printf("\n1. Add student\n");
    printf("2. Update marks\n");
    printf("3. Display marks\n");
    printf("Enter choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            add_student();
            break;
        case 2:
            update_marks();
            break;
        case 3:
            display_details();
            break;
        default:
```

```
            printf("Invalid input\n");
    }
}


void sort_database() {

    struct Student students[100];
    int i, j;
    struct Student temp;


    file_pointer = fopen("student.txt", "rb");
    if (file_pointer == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    i = 0;
    while (fread(&students[i], sizeof(struct Student), 1, file_pointer) == 1)
{
        i++;
    }
    student_count = i;

    fclose(file_pointer);


    for (i = 0; i < student_count - 1; i++) {
        for (j = 0; j < student_count - i - 1; j++) {
            if (students[j].totalmarks < students[j + 1].totalmarks) {
                temp = students[j];
                students[j] = students[j + 1];
                students[j + 1] = temp;
            }
        }
    }

    file_pointer = fopen("student.txt", "wb");
    if (file_pointer == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    for (i = 0; i < student_count; i++) {
        fwrite(&students[i], sizeof(struct Student), 1, file_pointer);
    }

    fclose(file_pointer);
```

```c
    printf("Database sorted by total marks.\n");
}

void student_menu() {

    sort_database();
    display_details();
}
int main() {
    int choice;
    while (1) {
        printf("\n1. Teacher\n");
        printf("2. Student\n");
        printf("3. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                teacher_menu();
                break;
            case 2:
                student_menu();
                break;
            case 3:
                exit(0);
            default:
                printf("Invalid input\n");
        }
    }
    return 0;
}
```

```
[be2288@localhost 2]$ git init
Initialized empty Git repository in /home/usr/student/ug/yr22/be2288/secondyear/se/gitassmt/2/.git/
[be2288@localhost 2]$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       2.c
nothing added to commit but untracked files present (use "git add" to track)
[be2288@localhost 2]$ git add 2.c
[be2288@localhost 2]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   2.c
#
```

```
[be2288@localhost 2]$ git commit -m "development complete"
[master (root-commit) 245e0aa] development complete
 1 file changed, 212 insertions(+)
 create mode 100644 2.c
```

```
1. Teacher
2. Student
3. Exit
Enter choice:
```

```
Enter choice: 1

1. Add student
2. Update marks
3. Display marks
Enter choice: 3
sh: cls: command not found

---Student details----
Rollno     Name            Math  CN    GT    OOS   SE    Total Marks
1          rohan           43    45    43    76    54    261
2          vikram          54    34    76    45    76    285
3          sunil           43    54    76    45    65    283
```

```
1. Teacher
2. Student
3. Exit
Enter choice: 2
Database sorted by total marks.
sh: cls: command not found

---Student details----
Rollno     Name            Math  CN    GT    OOS   SE    Total Marks
2          vikram          54    34    76    45    76    285
3          sunil           43    54    76    45    65    283
1          rohan           43    45    43    76    54    261
```

```
[be2288@localhost 2]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   2.c
#       new file:   student.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       a.out
[be2288@localhost 2]$ git commit -m "database save"
[master b6e73b6] database save
 2 files changed, 1 insertion(+)
 create mode 100644 student.txt
```

```
[be2288@localhost 2]$ git log
commit b6e73b6f97cc51371fac3f1a00890d71b8b6892b
Author: hrithvik kondalkar <kondalkarhrithvik@gmail.com>
Date:   Sat Apr 6 20:12:08 2024 +0530

    database save

commit 245e0aad88ba86e4f640451894784e6b9ce2b8ed
Author: hrithvik kondalkar <kondalkarhrithvik@gmail.com>
Date:   Sat Apr 6 20:03:54 2024 +0530

    development complete
```

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.

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

#define MAX_TASKS 100
#define MAX_DESCRIPTION_LENGTH 100

struct Task {
    char description[MAX_DESCRIPTION_LENGTH];
    bool completed;
};

void displayTasks(const struct Task* tasks, size_t size) {
    printf("Task List:\n");
    for (size_t i = 0; i < size; ++i) {
        printf("[%zu] %s %s\n", i + 1, tasks[i].completed ? "[Completed]" :
"[Pending]", tasks[i].description);
    }
}

void addTask(struct Task* tasks, size_t* size, const char* description) {
    if (*size >= MAX_TASKS) {
        printf("Error: Task list is full.\n");
        return;
    }
    struct Task newTask = {.completed = false};
    strncpy(newTask.description, description, MAX_DESCRIPTION_LENGTH - 1);
    tasks[(*size)++] = newTask;
}

void editTask(struct Task* tasks, size_t size, size_t taskIndex, const char*
newDescription) {
    if (taskIndex < size) {
        strncpy(tasks[taskIndex].description, newDescription,
MAX_DESCRIPTION_LENGTH - 1);
    } else {
        printf("Invalid task index.\n");
    }
}

void completeTask(struct Task* tasks, size_t size, size_t taskIndex) {
    if (taskIndex < size) {
        tasks[taskIndex].completed = true;
```

```
    } else {
        printf("Invalid task index.\n");
    }
}

void saveTasks(const struct Task* tasks, size_t size, const char* filename) {
    FILE* file = fopen(filename, "w");
    if (file != NULL) {
        for (size_t i = 0; i < size; ++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* size, const char* filename) {
    FILE* file = fopen(filename, "r");
    if (file != NULL) {
        *size = 0;
        while (fscanf(file, "%d", &tasks[*size].completed) != EOF) {
            fscanf(file, "%s", tasks[*size].description);
            (*size)++;
        }
        fclose(file);
    } else {
        printf("Error: Unable to load tasks from file.\n");
    }
}


int main() {
    struct Task tasks[MAX_TASKS];
    size_t size = 0;
    const char* filename = "task_history.txt";
    loadTasks(tasks, &size, 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);
        switch (choice) {
```

```
            case 1:
                displayTasks(tasks, size);
                break;
            case 2:
                {
                    char description[MAX_DESCRIPTION_LENGTH];
                    printf("Enter task description: ");
                    scanf(" %[^\n]", description);
                    addTask(tasks, &size, description);
                    break;
                }
            case 3:
                {
                    size_t taskIndex;
                    char newDescription[MAX_DESCRIPTION_LENGTH];
                    printf("Enter task index to edit: ");
                    scanf("%zu", &taskIndex);
                    printf("Enter new task description: ");
                    scanf(" %[^\n]", newDescription);
                    editTask(tasks, size, taskIndex - 1, newDescription);
                    break;
                }
            case 4:
                {
                    size_t taskIndex;
                    printf("Enter task index to mark as completed: ");
                    scanf("%zu", &taskIndex);
                    completeTask(tasks, size, taskIndex - 1);
                    break;
                }
            case 5:
                saveTasks(tasks, size, filename);
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

```
[be2288@localhost 3]$ vim 3.c
[be2288@localhost 3]$ [be2288@localhost 3]$ git init
Initialized empty Git repository in /home/usr/student/ug/yr22/be2288/secondyear/se/gitassmt/3/.git/
[be2288@localhost 3]$ git add 3.c
[be2288@localhost 3]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   3.c
#
```

```
Options:
1. Display Tasks
2. Add Task
3. Edit Task
4. Complete Task
5. Save and Quit
Enter your choice: 1
Task List:
[1] [Pending] complete homework
[2] [Pending] clean room
```

```
[be2288@localhost 3]$ git add *.txt
[be2288@localhost 3]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   3.c
#       new file:   task_history.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       a.out
[be2288@localhost 3]$ git commit -m "database save"
[master 0d1dbd7] database save
 2 files changed, 6 insertions(+), 2 deletions(-)
 create mode 100644 task_history.txt
```

```
[be2288@localhost 3]$ git log
commit 0d1dbd72a8ac3f3c10645003c9aecdf6a835d233
Author: hrithvik kondalkar <kondalkarhrithvik@gmail.com>
Date:   Sat Apr 6 20:26:12 2024 +0530

    database save

commit b17e184b7303a190011a6c803d4efaebd1784d9a
Author: hrithvik kondalkar <kondalkarhrithvik@gmail.com>
Date:   Sat Apr 6 20:17:00 2024 +0530

    development complete
```