SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of india)
Re-Accredited by NAAC with 'A++' Grade
॥वसुधैव कुटुम्बकम्॥

Assignment 12

Mini Project: Product Ranking System.

Name: Amit pawar

PRN:25070126501

Aiml-A1

Branch & Year: B.Tech (Aritifcial Intelligence & Machine Learning) 2rd Year.

College: Symbiosis Institute of Technology (SIT),pune

Section: A

Aim: This project is to develop a Product Ranking System using the C language. It enables users to sort products based on their rating, price, or sales with the help of the Bubble Sort algorithm.
The system also allows applying discounts to update product prices easily.
the project demonstrates the practical use of structures, arrays, and sorting techniques in C programming**.**

Theory:

The Product Ranking System is a mini project developed in the C programming language that demonstrates the real-world use of data structures and sorting algorithms.
The main goal of this project is to help users organize and manage products efficiently by sorting them based on rating, price, or sales (popularity).

In this project, details such as the name, price, rating, and number of sales of each product are stored using a structure.
A structure in C is a user-defined data type that allows different data elements to be grouped together under one name, making it easier to handle complex information.
An array of structures is used to store multiple product records, enabling easy sorting and display operations.

To organize the products meaningfully, the system uses the Bubble Sort algorithm.
This algorithm repeatedly compares and swaps adjacent elements if they are in the wrong order until the list becomes sorted.
The products can be sorted by rating (high to low), price (low to high), or sales (high to low).
This helps users easily identify the best-rated, most affordable, or most popular products.

An additional feature of this system is the discount function, which allows users to apply a percentage discount to all products.
After applying the discount, the program automatically updates and displays the new prices, similar to how e-commerce websites handle seasonal sales or promotions.

The program follows a menu-driven approach, giving users simple options to sort products, view the product list, apply discounts, or exit.
This design makes the program interactive, user-friendly, and easy to operate.

Methodology

The Product Ranking System project is developed using the C programming language and follows a step-by-step structured programming approach. The methodology includes several stages — from data collection to output generation ensuring that the program runs efficiently and achieves its objective of ranking and managing products.

1. Problem Definition

The main problem addressed by this project is the difficulty in managing and organizing a list of products based on multiple criteria such as rating, price, and sales.
The aim is to build a system that can sort products, display them in order, and apply discounts easily, similar to how e-commerce platforms rank and update product listings.

2. System Design

The system is designed using the structured programming paradigm of C, where the program is divided into smaller, manageable functions.
Key design components include:

- Structure Definition (struct Product) – used to store details like product name, price, rating, and sales.
- Array of Structures – used to hold multiple product records for sorting and manipulation.
- Functions – created for specific tasks such as sorting, displaying products, and applying discounts.

This modular approach improves readability, maintainability, and reusability of the code.

3. Data Storage and Representation

All product details are stored in a structure array, where:

- Each structure represents one product.
- The array holds all products entered by the user.

This method allows easy traversal, sorting, and updates to product data during program execution.

4. Sorting Mechanism

The core functionality of this project is sorting products using the Bubble Sort algorithm.
The sorting process works as follows:

1. The user selects a sorting criterion — rating, price, or sales.
2. The program repeatedly compares adjacent product records.
3. If the pair is out of order (based on the selected criterion), they are swapped.
4. This continues until the entire list is sorted.

Sorting options provided:

- **By Rating:** High to Low
- **By Price:** Low to High
- **By Sales:** High to Low

This helps in efficiently organizing products for better comparison and selection.

5. Discount Application

An additional feature is the discount module, where users can apply a percentage discount to all products.
Steps include:

1. The user enters a discount percentage.
2. The program calculates the reduced price using the formula:
   New Price = Original Price - (Original Price × Discount / 100)
3. The updated prices are displayed immediately. This simulates real-world discount systems used in e-commerce applications.

6. User Interaction (Menu-Driven Interface)

The project follows a menu-driven interface for ease of use.
Users can repeatedly choose from:

- Sorting products
- Displaying the product list
- Applying discounts
- Exiting the system

This ensures interactivity and allows users to perform multiple operations in a single session without restarting the program.

7. Implementation Tools

- Programming Language: C
- Compiler: GCC, Terminal
- Data Structure Used: Structure and Array
- Algorithm Used: Bubble Sort

8. Output Generation

After performing the selected operations, the program displays a neatly formatted product table showing:

- Product name
- Price (updated if discount applied)
- Rating
- Sales

This provides a clear and organized view of product information to the user.

Code: Product Ranking System.

```
// Amit Pawar
// AIML - A1
// Mini Project: Product Ranking System using Bubble Sort with Discount Feature
#include <stdio.h>
#include <string.h>
// Structure Definition
// Stores all details of a product
struct Product {
    char name[50];
    float price;
    float rating;
    int sales;
};

// Function to swap two products
void swap(struct Product *a, struct Product *b) {
    struct Product temp = *a;
    *a = *b;
    *b = temp;
}

// using Bubble Sort Function
void bubbleSort(struct Product arr[], int n, int choice) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {

            int swapFlag = 0;  // check if swapping is needed
                // Rating (High to Low)
            if (choice == 1 && arr[j].rating < arr[j + 1].rating)
                swapFlag = 1;  // Sort by rating (descending)

                // Price (Low to High)
            else if (choice == 2 && arr[j].price > arr[j + 1].price)
                swapFlag = 1;  // Sort by price (ascending)

                //  Sales (High to Low)
            else if (choice == 3 && arr[j].sales < arr[j + 1].sales)
```

```
            swapFlag = 1;  // Sort by sales (descending)

        if (swapFlag)
            swap(&arr[j], &arr[j + 1]);
      }
   }
}
// Function to Display All Products
void displayProducts(struct Product arr[], int n) {
   printf("\n%-15s %-10s %-10s %-10s\n", "Product", "Price", "Rating",
"Sales");
   printf("-------------------------------------\n");

   for (int i = 0; i < n; i++) {
      printf("%-15s %-10.2f %-10.1f %-10d\n",
           arr[i].name, arr[i].price, arr[i].rating, arr[i].sales);
   }
}

// Function to Apply Discount
// Reduces the price of all products by a given percentage
void applyDiscount(struct Product arr[], int n) {
   float discount;
   printf("\nEnter discount percentage (e.g., 10 for 10%%): ");
   scanf("%f", &discount);

   for (int i = 0; i < n; i++) {
      arr[i].price = arr[i].price - (arr[i].price * discount / 100);
   }

   printf("\n Discount of %.2f%% applied successfully!\n", discount);
   printf("\nUpdated product prices:\n");
   displayProducts(arr, n);
}

// Main Function
int main() {
   int n, choice;

   // number of products
```

```
printf("Enter number of products: ");
scanf("%d", &n);

struct Product products[n];

//  Input details of each product
for (int i = 0; i < n; i++) {
   printf("\nEnter details of product %d\n", i + 1);
   printf("Name: ");
   scanf("%s", products[i].name);
   printf("Price: ");
   scanf("%f", &products[i].price);
   printf("Rating (1-5): ");
   scanf("%f", &products[i].rating);
   printf("Sales (number of purchases): ");
   scanf("%d", &products[i].sales);
}

// Step 3: Menu-driven operations
while (1) {
   printf("\n============================");
   printf("\n     PRODUCT MENU");
   printf("\n============================");
   printf("\n1. Sort Products");
   printf("\n2. Show Product List");
   printf("\n3. Apply Discount");
   printf("\n4. Exit");
   printf("\nEnter your choice: ");
   scanf("%d", &choice);

   if (choice == 1) {
      int sortChoice;
      printf("\nSort products by:\n");
      printf("1. Rating (High to Low)\n");
      printf("2. Price (Low to High)\n");
      printf("3. Sales (High to Low)\n");
      printf("Enter your choice: ");
      scanf("%d", &sortChoice);
      bubbleSort(products, n, sortChoice);
      printf("\n--- Sorted Product List ---\n");
```

```
            displayProducts(products, n);
        }
        else if (choice == 2) {
            displayProducts(products, n);
        }
        else if (choice == 3) {
            applyDiscount(products, n);
        }
        else if (choice == 4) {
            printf("\nThank you for using the Product Ranking System! \n");
            break;
        }
        else {
            printf("\n Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

Output:

```
Enter number of products: 3

Enter details of product 1
Name: Laptop
Price: 55000
Rating (1-5): 4.5
Sales (number of purchases): 120


Enter details of product 2
Name: Headphones
Price: 2500
Rating (1-5): 4.2
Sales (number of purchases): 300
```

```
Enter details of product 3
Name: Mobile
Price: 20000
Rating (1-5): 4.8
Sales (number of purchases): 500


============================
   PRODUCT MENU
============================
1. Sort Products
2. Show Product List
3. Apply Discount
4. Exit
Enter your choice: 1

Sort products by:
1. Rating (High to Low)
2. Price (Low to High)
3. Sales (High to Low)
Enter your choice: 1

--- Sorted Product List ---
Product       Price     Rating    Sales
-------------------------------------
Mobile        20000.00  4.8       500
Laptop        55000.00  4.5       120
Headphones    2500.00   4.2       300


============================
   PRODUCT MENU
============================
1. Sort Products
2. Show Product List
3. Apply Discount
4. Exit
```

```
Enter your choice: 3


Enter discount percentage (e.g., 10 for 10%): 10

 Discount of 10.00% applied successfully!

Updated product prices:

Product        Price      Rating     Sales
-------------------------------------
Mobile         18000.00   4.8        500
Laptop         49500.00   4.5        120
Headphones     2250.00    4.2        300


============================
    PRODUCT MENU
============================
1. Sort Products
2. Show Product List
3. Apply Discount
4. Exit
Enter your choice: 4

Thank you for using the Product Ranking System!
```

Result:

The Product Ranking System was successfully designed and executed using the C programming language.
The program efficiently accepts product details such as name, price, rating, and sales, stores them using a structure array, and performs sorting operations based on user choice — rating (high to low), price (low to high), or sales (high to low) using the Bubble Sort algorithm.

After sorting, the system displays the updated product list in a well-formatted table, helping users easily identify the best-rated, most affordable, or most popular products.
The discount feature works accurately, updating all product prices according to the user-specified discount percentage.

All functions performed as expected during testing — sorting accuracy was verified, and price updates were calculated correctly after discount application.
The user-friendly, menu-driven interface ensures easy navigation and interaction throughout the program.

Hence, the project successfully meets its objectives by demonstrating how data structures and sorting algorithms can be applied to develop a practical and efficient product management system.

Discussion on Result:

The Product Ranking System works efficiently, accurately sorting products based on rating, price, or sales using the Bubble Sort algorithm.
The discount feature updates product prices correctly after applying the entered percentage.
Results show that the program is user-friendly, displays clear output, and meets its goal of ranking and managing products effectively using structures and sorting techniques in C.