PRODUCT RATING SYSTEM

Research:

Sources-

Youtube, Google websites- geeksforgeeks.org, www.programiz.com, www.w3schools.com, www.tutorialspoint.com

Analysis:

This C program calculates the percentage of marks from five subjects and determines pass/fail status along with a corresponding grade.

Key Components

1. **Variables**:
   - Marks for five subjects (phy, chem, eng, math, comp_sci).
   - float per for percentage calculation.
2. **Input Handling**:
   - Prompts for five integer marks.
   - Validates that exactly five marks are entered and that each mark is between 0 and 100.
3. **Percentage Calculation**:
   - Computes the percentage as the sum of marks divided by 5.0.
4. **Pass/Fail and Grading**:
   - If the percentage is 40 or above, the student passes and receives a grade (A, B, C, D, or E).
   - If below 40, the status is "FAIL."

Strengths

- Validates input to ensure marks are in the correct range.
- Clear output for percentage and status..

This program effectively meets its goals but can be improved for flexibility and user experience.

Ideate:

Concept Applications:

Product Rating System: To read multiple product name by user and create a list while displaying summary. Give user the ability to give star ratings for the listed products. User can also give product reviews in then form of comments (optional) that'll also be displayed in the summary. This code is used to determine if the listed product is qualified for market sale based on user feedback.

Build:

Algorithm (for base code):

1. Start

2. Initialize Variables

- Declare integer variables: phy, chem, eng, math, comp_sci
- Declare float variable: per
- Declare integer variable: validInput and set it to 1

3. Input Marks

- Display the message: "Enter five subjects marks (0-100):"
- Read five integers for phy, chem, eng, math, and comp_sci using scanf
- If the number of inputs read is not equal to 5, then:
    - Print: "Error: Please enter exactly five integer marks."
    - Exit the program with a status of 1

4. Validate Marks

- Check if each mark is within the range 0 to 100
- If any mark is out of this range, then:
    - Print: "Error: Marks should be between 0 and 100."
    - Exit the program with a status of 1

5. Calculate Percentage

- Calculate the total marks: total = phy + chem + eng + math + comp_sci
- Compute percentage: per = total / 5.0
- Print the percentage formatted to two decimal places: "Percentage = %.2f"

6. Determine Pass/Fail Status

- If per >= 40, then:
    - Print: "Status: PASS"
    - If per >= 90, then:

```
            ▪ Print: "Grade: A"
        o Else if per >= 80, then:
            ▪ Print: "Grade: B"
        o Else if per >= 70, then:
            ▪ Print: "Grade: C"
        o Else if per >= 60, then:
            ▪ Print: "Grade: D"
        o Else:
            ▪ Print: "Grade: E"
    • Else:
        o Print: "Status: FAIL"

7. End
```

Base Code:

```c
#include <stdio.h>

int main() {

  int phy, chem, eng, math, comp_sci;

  float per;

  int validInput = 1; // Flag for input validation

  printf("Enter five subjects marks (0-100): ");

  // Read the marks

  if (scanf("%d%d%d%d%d", &phy, &chem, &eng, &math, &comp_sci) != 5) {

    printf("Error: Please enter exactly five integer marks.\n");

    return 1; // Exit if input is not as expected

  }

  // Check if all marks are within the valid range

  if (phy < 0 || phy > 100 || chem < 0 || chem > 100 ||

    eng < 0 || eng > 100 || math < 0 || math > 100 ||

    comp_sci < 0 || comp_sci > 100) {
```

```
    printf("Error: Marks should be between 0 and 100.\n");
    return 1; // Exit if marks are out of range
}
// Calculate percentage
per = (phy + chem + eng + math + comp_sci) / 5.0;
printf("Percentage = %.2f\n", per);
// Determine pass/fail status and grade
if (per >= 40) {
    printf("Status: PASS\n");
    if (per >= 90) {
        printf("Grade: A\n");
    } else if (per >= 80) {
        printf("Grade: B\n");
    } else if (per >= 70) {
        printf("Grade: C\n");
    } else if (per >= 60) {
        printf("Grade: D\n");
    } else {
        printf("Grade: E\n");
    }
} else {
    printf("Status: FAIL\n");
}
return 0;  }
```

Real Life Applications:

Product Rating System: To read product name, star ratings and reviews from user to qualify products for market sale.

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX_NAME_LENGTH 100

int is_valid_product_name(const char *name) {

  if (strlen(name) == 0) {

    return 0; // Invalid

  }

  for (int i = 0; name[i] != '\0'; i++) {

    if (isdigit(name[i])) {

      return 0; // Invalid

    }

  }

  return 1; // Valid

}

int is_valid_comment(const char *comment) {

  for (int i = 0; comment[i] != '\0'; i++) {

    if (isdigit(comment[i])) {

      return 0; // Invalid

    }

  }
```

```
    return 1; // Valid

}

void clear_input_buffer() {

    while (getchar() != '\n'); // Clear the input buffer

}

int main() {

    int max_products = 10;

    char **product_names = malloc(max_products * sizeof(char *));

    float *ratings = malloc(max_products * sizeof(float));

    char **comments = malloc(max_products * sizeof(char *));

    int count = 0;

    char more;

    if (!product_names || !ratings || !comments) {

        printf("Memory allocation failed!\n");

        return 1;

    }

    for (int i = 0; i < max_products; i++) {

        product_names[i] = malloc(MAX_NAME_LENGTH * sizeof(char));

        comments[i] = NULL; // Initialize comment pointers

        if (!product_names[i]) {

            printf("Memory allocation failed!\n");

            return 1;

        }

    }

    do {
```

```
if (count >= max_products) {
    printf("Maximum number of products reached.\n");
    break;
}
// Read product name
while (1) {
    printf("Enter the product name: ");
    fgets(product_names[count], MAX_NAME_LENGTH, stdin);
    product_names[count][strcspn(product_names[count], "\n")] = 0; // Remove newline
    if (!is_valid_product_name(product_names[count])) {
        printf("Error: Product name must not be empty and cannot contain numeric characters.\n");
        continue; // Skip to the next iteration
    }
    break; // Valid input, exit loop
}
// Loop for star rating input until valid
while (1) {
    printf("Assign a star rating (0-5) for the product: ");
    if (scanf("%f", &ratings[count]) != 1) {
        printf("Error: Invalid input. Please enter a numeric value.\n");
        clear_input_buffer();
        continue;
    }
    if (ratings[count] < 0 || ratings[count] > 5) {
        printf("Error: Star rating must be between 0 and 5.\n");
        continue; // Prompt for input again
    }
    break; // Valid input, exit loop
}
// Clear the input buffer before reading a comment
clear_input_buffer();
// Read comment
while (1) {
    printf("Enter a comment for the product (optional, no numeric values allowed): ");
    char temp_comment[1024]; // Temporary buffer to read the comment
    fgets(temp_comment, sizeof(temp_comment), stdin);
    temp_comment[strcspn(temp_comment, "\n")] = 0; // Remove newline
    if (!is_valid_comment(temp_comment)) {
        printf("Error: Comment must not contain numeric values.\n");
        continue; // Prompt for input again
    }
    // Allocate memory for the comment
    comments[count] = malloc(strlen(temp_comment) + 1);
    if (!comments[count]) {
        printf("Memory allocation failed!\n");
        return 1;
    }
    strcpy(comments[count], temp_comment); // Copy the comment
    break; // Valid input, exit loop
}
```

```
        // Display qualification message
        if (ratings[count] > 2) {
            printf("Product '%s' qualified for sale.\n", product_names[count]);
        } else {
            printf("Product '%s' not qualified for sale.\n", product_names[count]);
        }
        count++;
        // Ask if the user wants to continue
        do {
            printf("Would you like to enter another product? (y/n): ");
            more = getchar(); // Read next character
            clear_input_buffer(); // Clear any extra characters in the input buffer
            if (more == 'y' || more == 'Y' || more == 'n' || more == 'N') {
                break; // Valid input, exit loop
            }
            printf("Invalid input. Please enter 'y' or 'n'.\n");
        } while (1);
    } while (more == 'y' || more == 'Y');
    // Only display summary if valid products were entered
    if (count > 0) {
        printf("\nSummary of rated products:\n");
        for (int i = 0; i < count; i++) {
            printf("Product: %s, Rating: %.2f, Comment: %s\n", product_names[i], ratings[i], comments[i]);
        }
    } else {
        printf("No products were entered.\n");
    }
    // Free allocated memory
    for (int i = 0; i < max_products; i++) {
        free(product_names[i]);
        free(comments[i]); // Free each comment
    }
    free(product_names);
    free(ratings);
    free(comments);
    return 0;
}
```

Test:

This code was tested on various compilers like VS Studio, and online compilers like Programiz.com, w3schools.com. The following output was attained for different input cases.

OUTPUT-

```
Enter the product name: Dior Sauvage Elixir

Assign a star rating (0-5) for the product: 4

Enter a comment for the product (optional, no numeric values allowed): Sauvage Elixir is an all rounder fragrance by Dior. After spraying, the initial notes might seem a bit spicy and musky but it later on waters down to a cool and musky fragrance. The projection is more than decent and you can definitely use it as a daily fragrance. Highly recommended!

Product 'Dior Sauvage Elixir' qualified for sale.

Would you like to enter another product? (y/n): y

Enter the product name: Lenovo IdeaPad Gaming

Assign a star rating (0-5) for the product: 3
```

```
Enter a comment for the product (optional, no numeric values allowed): The Lenovo IdeaPad Gaming three is a decent, value for money Laptop. It's pretty good for casual gaming and can process other tasks at high speeds due to it's Ryzen five processor. Its four GB Vram let's you play almost all popular games. The only drawbacks are it's heavy weight and below average battery backup of two hours.

Product 'Lenovo IdeaPad Gaming' qualified for sale.

Would you like to enter another product? (y/n): n


Summary of rated products:

Product: Dior Sauvage Elixir, Rating: 4.00, Comment: Sauvage Elixir is an all rounder fragrance by Dior. After spraying, the initial notes might seem a bit spicy and musky but it later on waters down to a cool and musky fragrance. The projection is more than decent and you can definitely use it as a daily fragrance. Highly recommended!

Product: Lenovo IdeaPad Gaming, Rating: 3.00, Comment: The Lenovo IdeaPad Gaming is a decent, value for money Laptop. It's pretty good for casual gaming and can process other tasks at high speeds due to it's Ryzen processor. Its Vram let's you play almost all popular games. The only drawbacks are it's heavy weight and below average battery backup of two hours.
```

Implement:

Structure and Functionality

1. **Memory Management**:
   - The program dynamically allocates memory for product names, ratings, and comments using `malloc`. It ensures that memory allocation is checked for success, and it frees all allocated memory at the end of the program to avoid memory leaks.
2. **Validation Functions**:
   - The `is_valid_product_name` function checks that the product name is not empty and contains no digits.
   - The `is_valid_comment` function ensures comments do not contain numeric characters.
   - These functions return `1` (true) for valid inputs and `0` (false) for invalid inputs.
3. **Input Handling**:
   - The program uses `fgets` to read strings safely, which helps avoid buffer overflow issues.
   - It employs a function `clear_input_buffer` to clear the input buffer after reading input, which prevents issues when reading multiple inputs in succession.

    - For star ratings, it validates numeric input and ensures the value falls within the specified range (0-5).
4. **User Interaction**:
    - The program prompts users to enter details for products in a loop, allowing them to input multiple products until they reach the maximum limit or choose to stop.
    - After entering details for each product, the program displays whether the product qualifies for sale based on its rating.
5. **Summary Output**:
    - At the end of the input process, the program summarizes all entered products, displaying names, ratings, and comments.

The code effectively implements a simple product rating system with validation and basic user interaction. While it functions well within its defined parameters, there are several areas for improvement that could enhance performance, usability, and flexibility. These changes could make the program more robust and user-friendly, catering to a wider range of scenarios.

Group Members:-

Sanskruti H. Gudhe :B24CE1043

Shivtej C. Wagare :B24CE1041

Sakshi A. Badakh :B24CE1045

Shubham S. Dorake :B24CE1042

Soham D. Badarkhe :B24CE1040