1. **Sum of Two Numbers**
   Write a program that takes two integers as input and calculates their sum using a function. Pass the integers to the function using call by value.

Sol:
```
#include <stdio.h>

// Function to calculate the sum of two integers

int sum(int a, int b) {

    return a + b;

}

int main() {

    int num1, num2;



    // Take input for two integers

    printf("Enter two integers: ");

    scanf("%d %d", &num1, &num2);



    // Call the function to calculate their sum

    int result = sum(num1, num2);

    printf("The sum of %d and %d is: %d\n", num1, num2, result);



    return 0;

}
```

O/p: Enter two integers: 10 20

The sum of 10 and 20 is: 30

2. **Swap Two Numbers**
   Write a program to swap two numbers using a function. Observe and explain why the original numbers remain unchanged due to call by value.

Sol: #include <stdio.h>

// Function to swap two numbers (using call by value)

void swap(int a, int b) {

    int temp = a;

    a = b;

    b = temp;

    printf("Inside swap function: a = %d, b = %d\n", a, b);

}

int main() {

    int num1, num2;

    // Take input for two numbers

    printf("Enter two numbers: ");

    scanf("%d %d", &num1, &num2);

    // Call the swap function

    swap(num1, num2);

```
    // Original numbers remain unchanged due to call by value
    printf("After swap function: num1 = %d, num2 = %d\n", num1, num2);

    return 0;
}
```

O/p: Enter two numbers: 3 7

Inside swap function: a = 7, b = 3

After swap function: num1 = 3, num2 = 7

3. **Find Maximum of Two Numbers**
   Implement a function that takes two integers as arguments and returns the larger of the two. Demonstrate how the original values are not altered.

Sol:

```
#include <stdio.h>

int max(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}

int main() {
```

```
    int x, y;
    printf("Enter two integers: ");
    scanf("%d %d", &x, &y);


    int result = max(x, y);
    printf("The maximum of %d and %d is: %d\n", x, y, result);
    return 0;
}
```

O/p: Enter two integers: 12 17

The maximum of 12 and 17 is: 17

4. **Factorial Calculation**
   Create a function to compute the factorial of a given number passed to it. Ensure the original number remains unaltered.

Sol:

```
#include <stdio.h>

// Function to compute the factorial of a number
int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

```
int main() {
    int num;

    // Take input for the number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Call the function to compute the factorial
    int result = factorial(num);
    printf("The factorial of %d is: %d\n", num, result);

    return 0;
}
```

O/p: Enter a number: 5

The factorial of 5 is: 120

5. **Check Even or Odd**
   Write a program where a function determines whether a given integer is even or odd. The function should use call by value.

Sol:

```
#include <stdio.h>

// Function to check if a number is even or odd
void check_even_odd(int n) {
    if (n % 2 == 0) {
```

```
        printf("%d is even.\n", n);
    } else {
        printf("%d is odd.\n", n);
    }
}

int main() {
    int num;

    // Take input for the number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Call the function to check even or odd
    check_even_odd(num);

    return 0;
}
```

O/p: Enter a number: 4

4 is even.

6. **Calculate Simple Interest**
   Write a program that calculates simple interest using a function. Pass principal, rate, and time as arguments and return the computed interest.

Sol:
```
#include <stdio.h>

// Function to calculate simple interest

float calculate_interest(float principal, float rate, float time) {

    return (principal * rate * time) / 100;

}


int main() {

    float principal, rate, time;


    // Take input for principal, rate, and time

    printf("Enter principal, rate, and time: ");

    scanf("%f %f %f", &principal, &rate, &time);


    // Call the function to calculate interest

    float interest = calculate_interest(principal, rate, time);

    printf("The simple interest is: %.2f\n", interest);


    return 0;

}
```

O/p: Enter principal, rate, and time: 1000 5 2

The simple interest is: 100.00

7. **Reverse a Number**
   Create a function that takes an integer and returns its reverse. Demonstrate how call by value affects the original number.

Sol: #include <stdio.h>

// Function to reverse a number

```
int reverse_number(int n) {
    int reversed = 0;
    while (n != 0) {
        int digit = n % 10;
        reversed = reversed * 10 + digit;
        n /= 10;
    }
    return reversed;
}


int main() {
    int num;

    // Take input for the number
    printf("Enter a number: ");
```

```
    scanf("%d", &num);


    // Call the function to reverse the number
    int reversed = reverse_number(num);
    printf("The reversed number is: %d\n", reversed);


    return 0;
}
```

O/p:

Enter a number: 12342

The reversed number is: 24321

8. **GCD of Two Numbers**
   Write a function to calculate the greatest common divisor (GCD) of two numbers passed by value.

Sol:

```
#include <stdio.h>

// Function to calculate the greatest common divisor (GCD)
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
```

```
    return a;
}

int main() {
    int num1, num2;

    // Take input for two integers
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    // Call the function to calculate GCD
    int result = gcd(num1, num2);
    printf("The GCD of %d and %d is: %d\n", num1, num2, result);

    return 0;
}
```

O/p: Enter two integers: 12 13

The GCD of 12 and 13 is: 1

9. **Sum of Digits**
   Implement a function that computes the sum of the digits of a number passed as an argument.

Sol: #include <stdio.h>

```
// Function to calculate the sum of digits
int sum_of_digits(int n) {
    int sum = 0;
    while (n != 0) {
        sum += n % 10;
        n /= 10;
    }
    return sum;
}

int main() {
    int num;

    // Take input for the number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Call the function to calculate the sum of digits
    int sum = sum_of_digits(num);
```

```
    printf("The sum of the digits is: %d\n", sum);


    return 0;
}
```

Sol:

Enter a number: 4321465

The sum of the digits is: 25

10. **Prime Number Check**
    Write a program where a function checks if a given number is prime. Pass the number as an argument by value.

Sol:

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


// Function to check if a number is prime
bool is_prime(int n) {
    if (n <= 1) return false;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
```

```
    return true;
}

int main() {
    int num;

    // Take input for the number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Call the function to check if the number is prime
    if (is_prime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}
```

O/p:

Enter a number: 7

7 is a prime number.

11. **Fibonacci Sequence Check**
    Create a function that checks whether a given number belongs to the Fibonacci sequence. Pass the number by value.

Sol:
```
#include <stdio.h>

#include <stdbool.h>


// Function to check if a number belongs to the Fibonacci sequence

bool is_fibonacci(int n) {

    int x1 = 5 * n * n + 4;

    int x2 = 5 * n * n - 4;


    return (is_perfect_square(x1) || is_perfect_square(x2));

}


int is_perfect_square(int x) {

    int s = (int)sqrt(x);

    return (s * s == x);

}


int main() {

    int num;
```

```
    // Take input for the number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Call the function to check if the number belongs to the Fibonacci sequence
    if (is_fibonacci(num)) {
        printf("%d is a Fibonacci number.\n", num);
    } else {
        printf("%d is not a Fibonacci number.\n", num);
    }

    return 0;
}
```

O/p: Enter a number: 13

13 is a Fibonacci number.

12. **Quadratic Equation Solver**
    Write a function to calculate the roots of a quadratic equation ax2+bx+c=0ax^2 + bx + c = 0ax2+bx+c=0. Pass the coefficients a,b,a, b,a,b, and ccc as arguments.

Sol:

```
#include <stdio.h>
#include <math.h>
```

```
void solve_quadratic(int a, int b, int c) {
    int discriminant = b * b - 4 * a * c;
    if (discriminant > 0) {
        double root1 = (-b + sqrt(discriminant)) / (2 * a);
        double root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("The roots are: %.2f and %.2f\n", root1, root2);
    } else if (discriminant == 0) {
        double root = -b / (2 * a);
        printf("The root is: %.2f\n", root);
    } else {
        printf("No real roots.\n");
    }
}

int main() {
    int a, b, c;
    printf("Enter coefficients a, b, and c: ");
    scanf("%d %d %d", &a, &b, &c);

    solve_quadratic(a, b, c);
    return 0;
```

}

O/p:

Enter coefficients a, b, and c: 1 2 1

The root is: -1.00

13. **Binary to Decimal Conversion**
Implement a function to convert a binary number (passed as an integer) into its decimal equivalent.

Sol: #include <stdio.h>

// Function to convert binary to decimal

int binary_to_decimal(int binary) {

    int decimal = 0, base = 1, remainder;

    while (binary > 0) {

        remainder = binary % 10;

        decimal += remainder * base;

        binary /= 10;

        base *= 2;

    }

    return decimal;

}

```
int main() {
    int binary;

    // Take input for the binary number
    printf("Enter a binary number: ");
    scanf("%d", &binary);

    // Call the function to convert binary to decimal
    int decimal = binary_to_decimal(binary);
    printf("The decimal equivalent is: %d\n", decimal);

    return 0;
}
```

O/P: Enter a binary number: 1101

The decimal equivalent is: 13

14. Matrix Trace Calculation

Write a program where a function computes the trace of a 2x2 matrix (sum of its diagonal elements). Pass the matrix elements individually as arguments.

Sol:

```
#include <stdio.h>

// Function to compute the trace of a 2x2 matrix
```

```
int matrix_trace(int a11, int a12, int a21, int a22) {
    return a11 + a22;  // Sum of diagonal elements
}

int main() {
    int a11, a12, a21, a22;

    // Take input for the 2x2 matrix
    printf("Enter the elements of the 2x2 matrix:\n");
    printf("a11: ");
    scanf("%d", &a11);
    printf("a12: ");
    scanf("%d", &a12);
    printf("a21: ");
    scanf("%d", &a21);
    printf("a22: ");
    scanf("%d", &a22);

    // Call the function to calculate the trace
    int trace = matrix_trace(a11, a12, a21, a22);
    printf("The trace of the matrix is: %d\n", trace);
```

```
    return 0;
}
```

O/P:

Enter the elements of the 2x2 matrix:

a11: 2

a12: 1

a21: 3

a22: 4

The trace of the matrix is: 6

15. Palindrome Number Check

Create a function that checks whether a given number is a palindrome. Pass the number by value and return the result

Sol:

```
#include <stdio.h>


// Function to check if a number is a palindrome
int is_palindrome(int num) {
    int original_num = num;
    int reversed = 0, remainder;

    // Reverse the number
    while (num != 0) {
```

```
        remainder = num % 10;
        reversed = reversed * 10 + remainder;
        num /= 10;
    }

    // Check if the original number is equal to the reversed number
    if (original_num == reversed) {
        return 1; // The number is a palindrome
    } else {
        return 0; // The number is not a palindrome
    }
}

int main() {
    int num;

    // Take input for the number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Call the function to check if the number is a palindrome
```

```
    if (is_palindrome(num)) {
        printf("%d is a palindrome.\n", num);
    } else {
        printf("%d is not a palindrome.\n", num);
    }

    return 0;
}
```

O/p: Enter a number: 1221

1221 is a palindrome.

1. Unit Conversion for Manufacturing Processes

- **Input:** A floating-point value representing the measurement and a character indicating the conversion type (e.g., 'C' for cm-to-inches or 'I' for inches-to-cm).
- **Output:** The converted value.
- **Function:**

float convert_units(float value, char type);

Sol: #include <stdio.h>

// Function to convert units

float convert_units(float value, char type) {

    float converted_value;

```
    if (type == 'C') {
        // Convert cm to inches
        converted_value = value / 2.54;
    } else if (type == 'I') {
        // Convert inches to cm
        converted_value = value * 2.54;
    } else {
        // Invalid type
        printf("Invalid conversion type.\n");
        return -1;
    }

    return converted_value;
}

int main() {
    float value;
    char type;

    // Input the value and conversion type
    printf("Enter the value to convert: ");
```

```
    scanf("%f", &value);

    printf("Enter conversion type ('C' for cm to inches, 'I' for inches to cm): ");

    scanf(" %c", &type);  // note the space before %c to consume any previous newline character


    // Call the function and display the result

    float result = convert_units(value, type);

    if (result != -1) {

        printf("Converted value: %.2f\n", result);

    }


    return 0;

}
```

O/p: Enter the value to convert: 123

Enter conversion type ('C' for cm to inches, 'I' for inches to cm): C

Converted value: 48.43

Enter the value to convert: 123 2

Enter conversion type ('C' for cm to inches, 'I' for inches to cm): I

Converted value: 309.88

2. Cutting Material Optimization

- **Input:** Two integers: the total length of the raw material and the desired length of each piece.

- **Output:** The maximum number of pieces that can be cut and the leftover material.
- **Function:**

int calculate_cuts(int material_length, int piece_length);

Sol: #include <stdio.h>

int calculate_cuts(int material_length, int piece_length) {

    return material_length / piece_length;

}

int main() {

    printf("Max pieces that can be cut: %d\n", calculate_cuts(1000, 200)); // Example usage

    return 0;

}

O/p: Max pieces that can be cut: 5

3. Machine Speed Calculation

- **Input:** Two floating-point numbers: belt speed (m/s) and pulley diameter (m).
- **Output:** The RPM of the machine.
- **Function:**

float calculate_rpm(float belt_speed, float pulley_diameter);

Sol: #include <stdio.h>

#include <math.h>

float calculate_rpm(float belt_speed, float pulley_diameter) {

    return (belt_speed * 60) / (M_PI * pulley_diameter);

}

int main() {

    printf("RPM of the machine: %.2f\n", calculate_rpm(10, 2)); // Example usage

    return 0;

}

O/p:

RPM of the machine: 95.49

4. Production Rate Estimation

- **Input:** Two integers: machine speed (units per hour) and efficiency (percentage).
- **Output:** The effective production rate.
- **Function:**

int calculate_production_rate(int speed, int efficiency);

Sol: #include <stdio.h>

// Function prototype

```
int calculate_production_rate(int speed, int efficiency);

int main() {
    int speed, efficiency;

    // Ask the user for machine speed and efficiency
    printf("Enter the machine speed (units per hour): ");
    scanf("%d", &speed);

    printf("Enter the machine efficiency (percentage): ");
    scanf("%d", &efficiency);

    // Calculate and display the effective production rate
    int production_rate = calculate_production_rate(speed, efficiency);
    printf("The effective production rate is: %d units per hour.\n", production_rate);

    return 0;
}

// Function definition to calculate effective production rate
int calculate_production_rate(int speed, int efficiency) {
```

```
    // Efficiency is given as a percentage, so divide by 100 to get the effective rate
    return (speed * efficiency) / 100;
}
```

O/p: Enter the machine speed (units per hour): 123

Enter the machine efficiency (percentage): 6%

The effective production rate is: 7 units per hour.

5. Material Wastage Calculation

- **Input:** Two integers: total material length and leftover material length.
- **Output:** The amount of material wasted.
- **Function:**

```
int calculate_wastage(int total_length, int leftover_length);
```

Sol:

```
#include <stdio.h>

int calculate_wastage(int total_length, int leftover_length) {
    return total_length - leftover_length;
}

int main() {
    printf("Material wastage: %d\n", calculate_wastage(1000, 200));  // Example usage
    return 0;
}
```

O/p: Material wastage: 800

6. Energy Cost Estimation

- **Input:** Three floating-point numbers: power rating (kW), operating hours, and cost per kWh.
- **Output:** The total energy cost.
- **Function:**

float calculate_energy_cost(float power_rating, float hours, float cost_per_kwh);

Sol: #include <stdio.h>

// Function prototype

float calculate_energy_cost(float power_rating, float hours, float cost_per_kwh);

int main() {

float power_rating, hours, cost_per_kwh;

// Ask the user for the power rating, operating hours, and cost per kWh

printf("Enter the power rating of the machine (in kW): ");

scanf("%f", &power_rating);

printf("Enter the number of operating hours: ");

scanf("%f", &hours);

printf("Enter the cost per kWh: ");

scanf("%f", &cost_per_kwh);

```
    // Calculate and display the total energy cost
    float total_cost = calculate_energy_cost(power_rating, hours, cost_per_kwh);
    printf("The total energy cost is: %.2f\n", total_cost);

    return 0;
}

// Function definition to calculate the total energy cost
float calculate_energy_cost(float power_rating, float hours, float cost_per_kwh) {
    // Energy consumed = power rating (kW) * operating hours
    // Total cost = energy consumed (kWh) * cost per kWh
    float energy_consumed = power_rating * hours;
    return energy_consumed * cost_per_kwh;
}
```

O/p: Enter the power rating of the machine (in kW): 300

Enter the number of operating hours: 3

Enter the cost per kWh: 1000

The total energy cost is: 900000.00

7. Heat Generation in Machines

- **Input:** Two floating-point numbers: power usage (Watts) and efficiency (%).

- **Output:** Heat generated (Joules).
- **Function:**

float calculate_heat(float power_usage, float efficiency);

Sol: #include <stdio.h>

// Function prototype

float calculate_heat(float power_usage, float efficiency);

int main() {

    float power_usage, efficiency;

    // Ask the user for power usage (in Watts) and efficiency (%)

    printf("Enter the power usage of the machine (in Watts): ");

    scanf("%f", &power_usage);

    printf("Enter the efficiency of the machine (percentage): ");

    scanf("%f", &efficiency);

    // Calculate and display the heat generated

    float heat_generated = calculate_heat(power_usage, efficiency);

    printf("The heat generated is: %.2f Joules\n", heat_generated);

```
    return 0;
}

// Function definition to calculate the heat generated
float calculate_heat(float power_usage, float efficiency) {
    // Heat generated is the difference between the power used and the effective output power
    // The ineffective power is lost as heat, and it is calculated by the difference
    return power_usage * (1 - efficiency / 100);
}
```

O/p: Enter the power usage of the machine (in Watts): 350

Enter the efficiency of the machine (percentage): 4%

The heat generated is: 336.00 Joules

8. Tool Wear Rate Calculation

- **Input:** A floating-point number for operating time (hours) and an integer for material type (e.g., 1 for metal, 2 for plastic).
- **Output:** Wear rate (percentage).
- **Function:**

```
float calculate_wear_rate(float time, int material_type);
```

Sol:

```
#include <stdio.h>

float calculate_wear_rate(float time, int material_type) {
    float wear_rate = 0;
```

```
    if (material_type == 1) {
        wear_rate = 0.02 * time;  // Metal wear rate
    } else if (material_type == 2) {
        wear_rate = 0.05 * time;  // Plastic wear rate
    }
    return wear_rate;
}

int main() {
    printf("Tool wear rate: %.2f%%\n", calculate_wear_rate(50, 1));  // Example usage
    return 0;
}
```

O/p: Tool wear rate: 1.00%

9. Inventory Management

- **Input:** Two integers: consumption rate (units/day) and lead time (days).
- **Output:** Reorder quantity (units).
- **Function:**

```
int calculate_reorder_quantity(int consumption_rate, int lead_time);
```

Sol:

```
#include <stdio.h>

// Function prototype
```

```
int calculate_reorder_quantity(int consumption_rate, int lead_time);

int main() {
    int consumption_rate, lead_time;

    // Ask the user for consumption rate and lead time
    printf("Enter the consumption rate (units/day): ");
    scanf("%d", &consumption_rate);

    printf("Enter the lead time (in days): ");
    scanf("%d", &lead_time);

    // Calculate and display the reorder quantity
    int reorder_quantity = calculate_reorder_quantity(consumption_rate, lead_time);
    printf("The reorder quantity is: %d units.\n", reorder_quantity);

    return 0;
}

// Function definition to calculate reorder quantity
int calculate_reorder_quantity(int consumption_rate, int lead_time) {
```

```
    // Reorder quantity is calculated as the consumption rate multiplied by the lead time

    return consumption_rate * lead_time;

}
```

O/p: Enter the consumption rate (units/day): 200

Enter the lead time (in days): 4

The reorder quantity is: 800 units.

10. Quality Control: Defective Rate Analysis

- **Input:** Two integers: number of defective items and total batch size.
- **Output:** Defective rate (percentage).
- **Function:**

float calculate_defective_rate(int defective_items, int batch_size);

Sol: #include <stdio.h>

```
float calculate_defective_rate(int defective_items, int batch_size) {

    return ((float)defective_items / batch_size) * 100;

}


int main() {

    printf("Defective rate: %.2f%%\n", calculate_defective_rate(5, 100));  // Example usage

    return 0;

}
```

O/p: Defective rate: 5.00%

11. Assembly Line Efficiency

- **Input:** Two integers: output rate (units/hour) and downtime (minutes).
- **Output:** Efficiency (percentage).
- **Function:**

float calculate_efficiency(int output_rate, int downtime);

Sol: #include <stdio.h>

float calculate_efficiency(int output_rate, int downtime) {

    return ((float)output_rate / (output_rate + downtime)) * 100;

}

int main() {

    printf("Assembly line efficiency: %.2f%%\n", calculate_efficiency(100, 10)); // Example usage

    return 0;

}

O/p: Assembly line efficiency: 90.91%

12. Paint Coverage Estimation

- **Input:** Two floating-point numbers: surface area (m^2) and paint coverage per liter (m^2/liter).
- **Output:** Required paint (liters).

- **Function:**

float calculate_paint(float area, float coverage);

Sol: #include <stdio.h>

```
float calculate_paint(float area, float coverage) {
    return area / coverage;
}

int main() {
    printf("Required paint: %.2f liters\n", calculate_paint(500, 10));  // Example usage
    return 0;
}
```

O/p: Required paint: 50.00 liters

13. Machine Maintenance Schedule

- **Input:** Two integers: current usage (hours) and maintenance interval (hours).
- **Output:** Hours remaining for maintenance.
- **Function:**

int calculate_maintenance_schedule(int current_usage, int interval);

Sol: #include <stdio.h>

```
int calculate_maintenance_schedule(int current_usage, int interval) {
    return interval - (current_usage % interval);
}

int main() {
    printf("Maintenance schedule: %d hours left\n",
calculate_maintenance_schedule(450, 500));  // Example usage
    return 0;
}
```

O/p: Maintenance schedule: 50 hours left

14. Cycle Time Optimization

- **Input:** Two integers: machine speed (units/hour) and number of operations per cycle.
- **Output:** Optimal cycle time (seconds).
- **Function:**

```
float calculate_cycle_time(int speed, int operations);
```

Sol: #include <stdio.h>

```
// Function prototype
float calculate_cycle_time(int speed, int operations);

int main() {
```

```
    int speed, operations;

    // Ask the user for machine speed and number of operations per cycle
    printf("Enter the machine speed (units/hour): ");
    scanf("%d", &speed);

    printf("Enter the number of operations per cycle: ");
    scanf("%d", &operations);

    // Calculate and display the optimal cycle time
    float cycle_time = calculate_cycle_time(speed, operations);
    printf("The optimal cycle time is: %.2f seconds.\n", cycle_time);

    return 0;
}

// Function definition to calculate optimal cycle time
float calculate_cycle_time(int speed, int operations) {
    // Cycle time is the time taken for one complete cycle in seconds
    // First, calculate time per unit (in hours), then convert to seconds
    float time_per_unit = 1.0 / speed;  // time per unit in hours
```

```
    float time_per_cycle = time_per_unit * operations * 3600;  // convert to seconds
    return time_per_cycle;
}
```

O/p: Enter the machine speed (units/hour): 200

Enter the number of operations per cycle: 5

The optimal cycle time is: 90.00 seconds.

1. Write a function that takes the original price of an item and a discount percentage as parameters. The function should return the discounted price without modifying the original price.

Function Prototype:

void calculateDiscount(**float** originalPrice, **float** discountPercentage);

Sol: #include <stdio.h>

```
// Function prototype
void calculateDiscount(float originalPrice, float discountPercentage);

int main() {
    float originalPrice, discountPercentage;

    // User input for original price and discount percentage
    printf("Enter the original price of the item: ");
```

```
    scanf("%f", &originalPrice);

    printf("Enter the discount percentage: ");
    scanf("%f", &discountPercentage);

    // Call the function to calculate and display the discounted price
    calculateDiscount(originalPrice, discountPercentage);

    return 0;
}

// Function definition
void calculateDiscount(float originalPrice, float discountPercentage) {
    // Calculate the discounted price
    float discountedPrice = originalPrice - (originalPrice * discountPercentage / 100);

    // Display the discounted price
    printf("The discounted price is: %.2f\n", discountedPrice);
}
```

O/p:

```
Enter the original price of the item: 200
```

Enter the discount percentage: 140

The discounted price is: -80.00

2. Create a function that takes the current inventory count of a product and a quantity to add or remove. The function should return the new inventory count without changing the original count.

Function Prototype:

int updateInventory(**int** currentCount, **int** changeQuantity);

Sol: #include <stdio.h>

```
// Function prototype
int updateInventory(int currentCount, int changeQuantity);

int main() {
    int currentCount, changeQuantity, newCount;

    // User input for current inventory count and quantity to change
    printf("Enter the current inventory count: ");
    scanf("%d", &currentCount);

    printf("Enter the quantity to add (positive) or remove (negative): ");
    scanf("%d", &changeQuantity);
```

```
    // Call the function to calculate the new inventory count
    newCount = updateInventory(currentCount, changeQuantity);

    // Display the new inventory count
    printf("The new inventory count is: %d\n", newCount);

    return 0;
}

// Function definition
int updateInventory(int currentCount, int changeQuantity) {
    // Calculate the new inventory count
    return currentCount + changeQuantity;
}
```

O/p: Enter the current inventory count: 60

Enter the quantity to add (positive) or remove (negative): 20

The new inventory count is: 80

Enter the current inventory count: 50

Enter the quantity to add (positive) or remove (negative): -20

The new inventory count is: 30

3. Implement a function that accepts the price of an item and a sales tax rate. The function should return the total price after tax without altering the original price.

Function Prototype:

float calculateTotalPrice(**float** itemPrice, **float** taxRate);

Sol: #include <stdio.h>

```
// Function prototype
float calculateTotalPrice(float itemPrice, float taxRate);

int main() {
    float itemPrice, taxRate, totalPrice;

    // User input for item price and tax rate
    printf("Enter the price of the item: ");
    scanf("%f", &itemPrice);

    printf("Enter the sales tax rate (as a percentage): ");
    scanf("%f", &taxRate);

    // Call the function to calculate the total price
    totalPrice = calculateTotalPrice(itemPrice, taxRate);
```

```
    // Display the total price
    printf("The total price after tax is: %.2f\n", totalPrice);

    return 0;
}

// Function definition
float calculateTotalPrice(float itemPrice, float taxRate) {
    // Calculate the total price after adding sales tax
    return itemPrice + (itemPrice * taxRate / 100);
}
```

O/p: Enter the price of the item: 190

Enter the sales tax rate (as a percentage): 10%

The total price after tax is: 209.00

4. Design a function that takes the amount spent by a customer and returns the loyalty points earned based on a specific conversion rate (e.g., 1 point for every $10 spent). The original amount spent should remain unchanged.

Function Prototype:

```
int calculateLoyaltyPoints(float amountSpent);
```

Sol: #include <stdio.h>

```
// Function prototype
int calculateLoyaltyPoints(float amountSpent);

int main() {
    float amountSpent;
    int loyaltyPoints;

    // Prompt the user for the amount spent
    printf("Enter the amount spent by the customer: ");
    scanf("%f", &amountSpent);

    // Calculate the loyalty points
    loyaltyPoints = calculateLoyaltyPoints(amountSpent);

    // Display the loyalty points earned
    printf("Loyalty points earned: %d\n", loyaltyPoints);

    return 0;
}

// Function to calculate loyalty points
```

```
int calculateLoyaltyPoints(float amountSpent) {
    const int conversionRate = 10; // 1 point for every $10 spent


    // Calculate and return loyalty points
    return (int)(amountSpent / conversionRate);
}
```

O/p:

Enter the amount spent by the customer: 120

Loyalty points earned: 12

5. Write a function that receives an array of item prices and the number of items. The function should return the total cost of the order without modifying the individual item prices.

Function Prototype:

```
float calculateOrderTotal(float prices[], int numberOfItems);
```

Sol:

```
#include <stdio.h>


// Function prototype
float calculateOrderTotal(float prices[], int numberOfItems);


int main() {
    int numberOfItems, i;
```

```
// Prompt the user for the number of items
printf("Enter the number of items: ");
scanf("%d", &numberOfItems);

float prices[numberOfItems];

// Input the prices of the items
printf("Enter the prices of the items:\n");
for (i = 0; i < numberOfItems; i++) {
    printf("Item %d: ", i + 1);
    scanf("%f", &prices[i]);
}

// Calculate the total cost of the order
float totalCost = calculateOrderTotal(prices, numberOfItems);

// Display the total cost
printf("The total cost of the order is: %.2f\n", totalCost);

return 0;
```

```
}

// Function definition
float calculateOrderTotal(float prices[], int numberOfItems) {
    float total = 0;

    // Iterate through the array to calculate the total
    for (int i = 0; i < numberOfItems; i++) {
        total += prices[i];
    }

    return total;
}
```

O/p:

```
Enter the number of items: 4
Enter the prices of the items:
Item 1: 100
Item 2: 150
Item 3: 98
Item 4: 80
The total cost of the order is: 428.00
```

6. Create a function that takes an item's price and a refund percentage as input. The function should return the refund amount without changing the original item's price.

Function Prototype:

float calculateRefund(**float** itemPrice, **float** refundPercentage);

Sol: #include <stdio.h>

```
// Function prototype
float calculateRefund(float itemPrice, float refundPercentage);

int main() {
    float itemPrice, refundPercentage, refundAmount;

    // Prompt the user for the item's price and refund percentage
    printf("Enter the item's price: ");
    scanf("%f", &itemPrice);

    printf("Enter the refund percentage: ");
    scanf("%f", &refundPercentage);

    // Calculate the refund amount
```

```
    refundAmount = calculateRefund(itemPrice, refundPercentage);

    // Display the refund amount
    printf("The refund amount is: %.2f\n", refundAmount);

    return 0;
}

// Function definition
float calculateRefund(float itemPrice, float refundPercentage) {
    // Calculate and return the refund amount
    return (itemPrice * refundPercentage / 100);
}
```

O/p: Enter the item's price: 300

Enter the refund percentage: 20

The refund amount is: 60.00

7. Implement a function that takes the weight of a package and calculates shipping costs based on weight brackets (e.g., $5 for up to 5kg, $10 for 5-10kg). The original weight should remain unchanged.

Function Prototype:

float calculateShippingCost(**float** weight);

Sol:

```
#include <stdio.h>

// Function prototype
float calculateShippingCost(float weight);

int main() {
    float packageWeight, shippingCost;

    // Prompt the user for the weight of the package
    printf("Enter the weight of the package (in kg): ");
    scanf("%f", &packageWeight);

    // Calculate the shipping cost
    shippingCost = calculateShippingCost(packageWeight);

    // Display the shipping cost
    printf("The shipping cost is: $%.2f\n", shippingCost);

    return 0;
}
```

```
// Function definition
float calculateShippingCost(float weight) {
    // Determine the shipping cost based on weight brackets
    if (weight <= 5.0) {
        return 5.0; // $5 for up to 5kg
    } else if (weight <= 10.0) {
        return 10.0; // $10 for 5-10kg
    } else if (weight <= 20.0) {
        return 20.0; // $20 for 10-20kg
    } else {
        return 50.0; // $50 for over 20kg
    }
}
```

O/p: Enter the weight of the package (in kg): 200

The shipping cost is: $50.00

8. Design a function that converts an amount from one currency to another based on an exchange rate provided as input. The original amount should not be altered.

Function Prototype:

float convertCurrency(**float** amount, **float** exchangeRate);

Sol: #include <stdio.h>

```
// Function prototype
float convertCurrency(float amount, float exchangeRate);

int main() {
    float amount, exchangeRate, convertedAmount;

    // Prompt the user for the amount and exchange rate
    printf("Enter the amount to convert: ");
    scanf("%f", &amount);

    printf("Enter the exchange rate: ");
    scanf("%f", &exchangeRate);

    // Convert the currency
    convertedAmount = convertCurrency(amount, exchangeRate);

    // Display the converted amount
    printf("The converted amount is: %.2f\n", convertedAmount);

    return 0;
}
```

```
// Function definition
float convertCurrency(float amount, float exchangeRate) {
    // Calculate and return the converted amount
    return amount * exchangeRate;
}
```

O/p: Enter the amount to convert: 3 400

Enter the exchange rate: 100

The converted amount is: 40000.00

9. Write a function that takes two prices from different vendors and returns the lower price without modifying either input price.

Function Prototype:

float findLowerPrice(**float** priceA, **float** priceB);

Sol: #include <stdio.h>

```
// Function prototype
float findLowerPrice(float priceA, float priceB);

int main() {
    float priceA, priceB, lowerPrice;

    // Prompt the user for prices from two vendors
```

```
    printf("Enter the price from vendor A: ");
    scanf("%f", &priceA);

    printf("Enter the price from vendor B: ");
    scanf("%f", &priceB);

    // Find the lower price
    lowerPrice = findLowerPrice(priceA, priceB);

    // Display the lower price
    printf("The lower price is: %.2f\n", lowerPrice);

    return 0;
}

// Function definition
float findLowerPrice(float priceA, float priceB) {
    // Return the lower of the two prices
    return (priceA < priceB) ? priceA : priceB;
}
```

O/p: Enter the price from vendor A: 200

Enter the price from vendor B: 400

The lower price is: 200.00

10. Create a function that checks if a customer is eligible for a senior citizen discount based on their age. The function should take age as input and return whether they qualify without changing the age value.

Function Prototype:

bool isEligibleForSeniorDiscount(**int** age);

Sol: #include <stdio.h>

#include <stdbool.h>

// Function prototype

bool isEligibleForSeniorDiscount(int age);

int main() {

    int age;

    // Ask the user for their age

    printf("Enter the customer's age: ");

    scanf("%d", &age);

    // Check eligibility for senior discount

```
    if (isEligibleForSeniorDiscount(age)) {
        printf("The customer is eligible for a senior citizen discount.\n");
    } else {
        printf("The customer is not eligible for a senior citizen discount.\n");
    }

    return 0;
}

// Function definition
bool isEligibleForSeniorDiscount(int age) {
    // Senior discount eligibility is 65 years or older
    return age >= 65;
}
```

O/p: Enter the customer's age: 68

The customer is eligible for a senior citizen discount.