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.

```
 #include <stdio.h>
int sum(int a, int b) {
   return a + b;
}
int main() {
   int num1, num2;
   printf("Enter two integers: ");
   scanf("%d %d", &num1, &num2);
   int result = sum(num1, num2);
   printf("Sum: %d\n", result);
   return 0;
}
```

Output:

Enter two integers: 34 12

Sum: 46

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.

```
#include <stdio.h>
 void swap(int a,int b){
  int temp=a;
  a=b;
  b=temp;
  printf("After swapping:a=%d,b=%d\n",a,b);
 }
 int main()
 {
    int num1,num2;
    printf("Enter two numbers:");
```

```
    scanf("%d%d",&num1,&num2);
    printf("Before Swapping:a=%d,b=%d\n",num1,num2);
    swap(num1,num2);
    printf("Original value:a=%d,b=%d\n",num1,num2);
}
```

Output:

Enter two numbers:12 32

Before Swapping:a=12,b=32

After swapping:a=32,b=12

Original value:a=12,b=32

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.

```
#include <stdio.h>
int max(int x,int y){
 if(x>y)
  return x;
 else
  return y;
 }
int main(){
   int a,b;
   printf("Enter two numbers:");
   scanf("%d%d",&a,&b);
   int result=max(a,b);
   printf("The maximun num of %d and %d is:%d",a,b,result);
   return 0;
}
```

Output:

Enter two numbers:12 34

The maximun num of 12 and 34 is:34

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

```
#include <stdio.h>

int factorial(int n) {
   if (n == 0 || n == 1)
      return 1;
   else
      return n * factorial(n - 1);
}

int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   int result = factorial(num);
   printf("Factorial of %d is: %d\n", num, result);
   return 0;
}
```

Output:

```
Enter an integer: 7
Factorial of 7 is: 5040
```

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.

```
#include <stdio.h>

void check_even_odd(int num) {
   if (num % 2 == 0)
      printf("even");
   else
      printf("odd");
}
```

```
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   check_even_odd(num);
   return 0;
}
```

Output:

Enter an integer: 12

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.

```
#include <stdio.h>
float calculate_interest(float p, float r, float t) {
   return (p*r*t) / 100;
}
int main() {
   float p, r, t;
   printf("Enter principal, rate, and time: ");
   scanf("%f %f %f", &p, &r, &t);
   float interest = calculate_interest(p, r, t);
   printf("The simple interest is: %.2f\n", interest);
   return 0;
}
```

Output:

Enter principal, rate, and time: 12 14 43

The simple interest is: 72.24

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

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

```
int reverse_number(int  num) {
   int reversed = 0;
   while (num != 0) {
      reversed = reversed * 10 + num % 10;
      num /= 10;
   }
   return reversed;
}
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   int reversed = reverse_number(num);
   printf("Reversed number: %d\n", reversed);
   return 0;
}
```

Output:

Enter an integer: 23

Reversed number: 32

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

```
#include <stdio.h>
int gcd(int a, int b) {
   while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
   }
   return a;
}
```

```
int main() {
   int x, y;
   printf("Enter two integers: ");
   scanf("%d %d", &x, &y);
   int result = gcd(x, y);
   printf("GCD of %d and %d is: %d\n", x, y, result);
   return 0;
}
```

Output:

Enter two integers: 45 65

GCD of 45 and 65 is: 5

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

```
#include <stdio.h>
int sum_of_digits(int num) {
   int sum = 0;
   while (num != 0) {
      sum += num % 10;
      num /= 10;
   }
   return sum;
}
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   int result = sum_of_digits(num);
   printf("Sum of digits: %d\n", result);
   return 0;
}
```

Output:

Enter an integer: 78

Sum of digits: 15

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.

```
#include <stdio.h>

int is_prime(int num) {
   if (num <= 1) return 0;
   for (int i = 2; i * i <= num; i++) {
      if (num % i == 0)
         return 0;
   }
   return 1;
}

int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   if (is_prime(num))
      printf("%d is a prime number.\n", num);
   else
      printf("%d is not a prime number.\n", num);
   return 0;
}
```

Output:

Enter an integer: 78

78 is not 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.

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

```
int is_fibonacci(int num) {
   int a = 0, b = 1, next;
   while (b < num) {
      next = a + b;
      a = b;
      b = next;
   }
   return b == num;
}
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   if (is_fibonacci(num))
      printf(" Fibonacci");
   else
      printf( "not a fibonacci");
   return 0;
}
```

Output:

```
Enter an integer: 56
not a fibonacci
```

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.

```
#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;
}
```

Output:

Enter coefficients a, b, and c: 1 -3 1

The roots are: 2.62 and 0.38

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

```
#include <stdio.h>
#include <math.h>
int binary_to_decimal(int binary) {
    int decimal = 0, i = 0;
    while (binary != 0) {
        decimal += (binary % 10) * pow(2, i);
        i++;
        binary /= 10;
    }
```

```
    return decimal;
}
int main() {
    int binary;
    printf("Enter a binary number: ");
    scanf("%d", &binary);
    int decimal = binary_to_decimal(binary);
    printf("Decimal equivalent: %d\n", decimal);
    return 0;
}
```

Output:

Enter a binary number: 22.4

Decimal equivalent: 6

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.

```
#include <stdio.h>
int matrix_trace(int a11, int a22) {
    return a11 + a22;
}
int main() {
    int a11, a12, a21, a22;
    printf("Enter elements of 2x2 matrix: ");
    scanf("%d %d %d %d", &a11, &a12, &a21, &a22);

    int trace = matrix_trace(a11, a22);
    printf("Trace of the matrix: %d\n", trace);
    return 0;
}
```

Output:

Enter elements of 2x2 matrix: 3 5 8 9

Trace of the matrix: 12

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.

```
#include <stdio.h>
int is_palindrome(int  num) {
   int original = num, reversed = 0;
   while (num != 0) {
      reversed = reversed * 10 + num % 10;
      num /= 10;
   }
   return original == reversed;
}
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   if (is_palindrome(num))
      printf("it is a palindrome");
   else
      printf("It is not a palindrome");

   return 0;
}
```

Output:

```
Enter an integer: 43
It is not 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)**

```
float convert_units(float value, char type);
#include <stdio.h>
float convert_units(float value, char type) {
    if (type == 'C') {
        return value / 2.54;
    } else if (type == 'I') {
        return value * 2.54;
    }
    return -1;
}
int main() {
    printf("Converted value: %.2f\n", convert_units(100, 'C'));
    return 0;
}
```

Output:

Converted value: 39.37

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);
#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;
```

```
}
```

Output:

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);
#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));
   return 0;
}
```

Output:

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);
#include <stdio.h>
int calculate_production_rate(int speed, int efficiency) {
   return (speed * efficiency) / 100;
}
```

```
int main() {
    printf("Effective production rate: %d\n", calculate_production_rate(100, 80));
    return 0;
}
```

Output:

Effective production rate: 80

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);
#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));
    return 0;
}
```

Output:

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);
#include <stdio.h>
float calculate_energy_cost(float power_rating, float hours, float cost_per_kwh) {
    return power_rating * hours * cost_per_kwh;
}
```

```
int main() {
   printf("Energy cost: %.2f\n", calculate_energy_cost(5, 10, 0.1));
   return 0;
}
```

Output:

Energy cost: 5.00

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);
```

```
#include <stdio.h>
float calculate_heat(float power_usage, float efficiency) {
   return power_usage * (1 - (efficiency / 100));
}
int main() {
   printf("Heat generated: %.2f Joules\n", calculate_heat(500, 80));
   return 0;
}
```

Output:

Heat generated: 100.00 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);
```

```
#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;
}
```

Output:

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);
#include <stdio.h>
// Function prototype
int calculate_reorder_quantity(int consumption_rate, int lead_time);

int main() {
    int consumption_rate, 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;
}
```

Output:

```
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);
#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;
}
```

Output:

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);
#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;
}
```

Output:

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);
#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;
```

```
}
```

Output::

```
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);
#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;
}
```

Output:

```
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);
#include <stdio.h>
float calculate_cycle_time(int speed, int operations);
int main() {
    int speed, operations;
    printf("Enter the machine speed (units/hour): ");
```

```
    scanf("%d", &speed);
    printf("Enter the number of operations per cycle: ");
    scanf("%d", &operations);
    float cycle_time = calculate_cycle_time(speed, operations);
    printf("The optimal cycle time is: %.2f seconds.\n", cycle_time);

    return 0;
}
float calculate_cycle_time(int speed, int operations) {
    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);
#include <stdio.h>
void calculateDiscount(float originalPrice, float discountPercentage);
int main() {
    float originalPrice, discountPercentage;
    printf("Enter the original price of the item: ");
    scanf("%f", &originalPrice);
    printf("Enter the discount percentage: ");
    scanf("%f", &discountPercentage);
    calculateDiscount(originalPrice, discountPercentage);
```

```
    return 0;
}
void calculateDiscount(float originalPrice, float discountPercentage) {
    // Calculate the discounted price
    float discountedPrice = originalPrice - (originalPrice * discountPercentage / 100);
    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);
#include <stdio.h>
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);
    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) {
```

```
    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);
```

```
#include <stdio.h>
float calculateTotalPrice(float itemPrice, float taxRate);
int main() {
    float itemPrice, taxRate, totalPrice;
    printf("Enter the price of the item: ");
    scanf("%f", &itemPrice);
    printf("Enter the sales tax rate (as a percentage): ");
    scanf("%f", &taxRate);
    totalPrice = calculateTotalPrice(itemPrice, taxRate);
    printf("The total price after tax is: %.2f\n", totalPrice);
    return 0;
}
float calculateTotalPrice(float itemPrice, float taxRate) {
    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);
```

```
#include <stdio.h>

int calculateLoyaltyPoints(float amountSpent);

int main() {
    float amountSpent;
    int loyaltyPoints;
    printf("Enter the amount spent by the customer: ");
    scanf("%f", &amountSpent);
    loyaltyPoints = calculateLoyaltyPoints(amountSpent);
    printf("Loyalty points earned: %d\n", loyaltyPoints);

    return 0;
}

int calculateLoyaltyPoints(float amountSpent) {
    const int conversionRate = 10;
    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);
```

```
#include <stdio.h>

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

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

```
    printf("Enter the number of items: ");
    scanf("%d", &numberOfItems);
    float prices[numberOfItems];
    printf("Enter the prices of the items:\n");
    for (i = 0; i < numberOfItems; i++) {
        printf("Item %d: ", i + 1);
        scanf("%f", &prices[i]);
    }
    float totalCost = calculateOrderTotal(prices, numberOfItems);
    printf("The total cost of the order is: %.2f\n", totalCost);
    return 0;
}
float calculateOrderTotal(float prices[], int numberOfItems) {
    float total = 0;
    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);
```

```
#include <stdio.h>
float calculateRefund(float itemPrice, float refundPercentage);
int main() {
    float itemPrice, refundPercentage, refundAmount;
    printf("Enter the item's price: ");
    scanf("%f", &itemPrice);
    printf("Enter the refund percentage: ");
    scanf("%f", &refundPercentage);
    refundAmount = calculateRefund(itemPrice, refundPercentage);
    printf("The refund amount is: %.2f\n", refundAmount);
    return 0;
}
float calculateRefund(float itemPrice, float refundPercentage) {
    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);
#include <stdio.h>
float calculateShippingCost(float weight);
int main() {
    float packageWeight, shippingCost;
    printf("Enter the weight of the package (in kg): ");
    scanf("%f", &packageWeight);
    shippingCost = calculateShippingCost(packageWeight);
    printf("The shipping cost is: $%.2f\n", shippingCost);
    return 0;
```

```
}
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);
```

```
#include <stdio.h>
float convertCurrency(float amount, float exchangeRate);
int main() {
    float amount, exchangeRate, convertedAmount;
    printf("Enter the amount to convert: ");
    scanf("%f", &amount);

    printf("Enter the exchange rate: ");
    scanf("%f", &exchangeRate);
    convertedAmount = convertCurrency(amount, exchangeRate);
    printf("The converted amount is: %.2f\n", convertedAmount);
    return 0;
```

```
}
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);
#include <stdio.h>
float findLowerPrice(float priceA, float priceB);
int main() {
    float priceA, priceB, lowerPrice;
    printf("Enter the price from vendor A: ");
    scanf("%f", &priceA);

    printf("Enter the price from vendor B: ");
    scanf("%f", &priceB);
    lowerPrice = findLowerPrice(priceA, priceB);
    printf("The lower price is: %.2f\n", lowerPrice);
    return 0;
}
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);
```

```
#include <stdio.h>
#include <stdbool.h>
bool isEligibleForSeniorDiscount(int age);
int main() {
    int age;
    printf("Enter the customer's age: ");
    scanf("%d", &age);
    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;
}
bool isEligibleForSeniorDiscount(int age) {
    return age >= 65;
}
```

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

The customer is eligible for a senior citizen discount.