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.

   With return type:

```
#include <stdio.h>

int calculateSum(int, int);



int main() {

   int num1=20, num2=10, sum;

   sum = calculateSum(num1, num2);



   printf("The sum is: %d\n", sum);



   return 0;

}



/*

Name:calculateSum()

Return Type: int

Parameter:int a, int b (data type of each parameter): No parameters

Shord disciption: it is used to calculate sum of 2 integers

*/

int calculateSum(int a, int b) {

   a=100,b=50;

   return a + b;

}
```

Without return Type

```
#include<stdio.h>
void sum(int a, int b);
int main(){
    int n1,n2;
    sum(n1,n2);
    return 0;
}

void sum(int a, int b){
    a=50,b=100;
    int sum=0;
    sum=a+b;
    printf("%d+%d=%d\n",a,b,sum);
}
```

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.

Swap without return type

```
#include<stdio.h>
void swap(int a, int b);
int main(){
    int n1=10,n2=20;
    printf("n1=%d,n2=%d\n",n1,n2);
    swap(n1,n2);
    printf("n1=%d,n2=%d\n",n1,n2);
    //Value of n1 and n2 remains unchanged because the copy of variable is being passed to swap function,the changes affect only the local variables (a and b) within the function.

    return 0;

}

void swap(int a, int b){

    a=50,b=100;

    printf("a=%d,b=%d\n",a,b);

    int temp=a;

    a=b;

    b=temp;

    printf("a=%d,b=%d\n",a,b);

}
```

Swap with return type

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 findMaximum(int, int);

int main() {

    int num1 = 20, num2 = 10;

    int max = findMaximum(num1, num2);

    // Print the maximum value

    printf("The maximum value is: %d\n", max);

    // Show that the original values remain unchanged

    printf("In main() - num1: %d, num2: %d\n", num1, num2);
    return 0;
}


/*
Name: findMaximum
Return Type: int
Parameters: int a, int b
Short Description: This function compares two integers and returns the larger of the two.
*/
int findMaximum(int a, int b) {
     a=100,b=90;
    if (a > b)
        return a;
    else
        return b;
}
```

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

   Without Return type:

```
#include <stdio.h>
void factorial(int);
int main() {
    int num1;
    factorial(num1);
    return 0;
}

/*
Name: factorial()
Return Type: void
Parameters: int a
Short Description: This function factorial of a number
*/
void factorial(int a) {
    int factorial=1;
    printf("Enter a number:\n");
    scanf("%d",&a);
    for(int i=1;i<=a;i++){
        factorial*=i;
    }
    printf("%d!=%d\n",a,factorial);
}
```

With return type

```
#include <stdio.h>
int factorial(int);
int main() {
    int num1;
    printf("Enter a number:\n");
    scanf("%d",&num1);
```

```
  int fact= factorial(num1);

   printf("%d!=%d\n",num1,fact);

   return 0;

}


/*

Name: factorial()

Return Type: int

Parameters: int a

Short Description: This function factorial of a number

*/

int factorial(int a) {

   int factorial=1;

   for(int i=1;i<=a;i++){

       factorial*=i;

   }

  return factorial;

}
```

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.

   Without return:

```
#include <stdio.h>

// Function declaration

void checkEvenOdd(int);
```

```
int main() {
    int num;
    // Prompt the user for input
    printf("Enter a number: ");
    scanf("%d", &num);
    // Calling the function to check if the number is even or odd
    checkEvenOdd(num);
    return 0;
}

/*
Name: checkEvenOdd
Return Type: void
Parameters: int num
Short Description: This function checks whether a number is even or odd.
*/
void checkEvenOdd(int num) {
    if (num % 2 == 0) {
        printf("%d is Even\n", num);
    } else {
        printf("%d is Odd\n", num);
    }
}
```

With return

```
#include <stdio.h>

// Function declaration

int checkEvenOdd(int);


int main() {

   int num;

   // Prompt the user for input

   printf("Enter a number: ");

   scanf("%d", &num);

   // Calling the function to check if the number is even or odd

   if(checkEvenOdd(num))

   {

      printf("\nThe number is even\n");

      }

   else

   printf("Number is odd");

   return 0;

}


/*

Name: checkEvenOdd

Return Type: void

Parameters: int num

Short Description: This function checks whether a number is even or odd.
```

```
*/
int checkEvenOdd(int num) {
    if (num % 2 == 0) {
        return 1;
    } else {
       return 0;
    }
}
```

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>
int SIN(int,int,int);
int main(){
    int principal,rate,year;
    printf("Enter principal amount\n");
    scanf("%d",&principal);
    printf("Enter rate of interest\n");
    scanf("%d",&rate);
     printf("Enter duration\n");
    scanf("%d",&year);
    int simpleInterest=SIN(principal,rate,year);
    printf("Simple Interest=%d\n",simpleInterest);
    return 0;
}
```

```
/* Name: SIN
Return Type: int
Parameters: int principal, int rate, int time
Short Description: This function calculates the simple interest for the given principal, rate, and time.
*/
int SIN(int p,int r, int y){
    int SI=(p*r*y)/100;
    return SI;


}
```

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 reverseNumber(int);
int main() {
    int num, reversed;
    printf("Enter an integer: ");
    scanf("%d", &num);
    reversed = reverseNumber(num);
    printf("The reverse of %d is: %d\n", num, reversed);
    return 0;
}


/*
Name: reverseNumber
```

```
Return Type: int
Parameters: int num
Short Description: This function reverses the digits of the given number.
*/
int reverseNumber(int num) {
    int reversed = 0;
    int digit;
    // Reversing the number
    while (num != 0) {
        digit=num%10;
        reversed = reversed * 10 + digit;
        num /= 10;
    }

    return reversed;
}
```

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 calculateGCD(int, int);
int main() {
    int num1, num2, gcd;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);

    gcd = calculateGCD(num1, num2);



    // Output the result

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



    return 0;

}



/*

Name: calculateGCD

Return Type: int

Parameters: int a, int b

Short Description: This function calculates the Greatest Common Divisor (GCD) of
two numbers using the Euclidean algorithm.

*/

int calculateGCD(int a, int b) {

    // Applying the Euclidean algorithm

    while (b != 0) {

        int temp = b;

        b = a % b;

        a = temp;

    }

    return a; // The GCD will be stored in 'a'

}
```

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 sumOfDigits(int);



int main() {

    int num, sum;





    printf("Enter a number: ");

    scanf("%d", &num);

    sum = sumOfDigits(num);



    // Output the result

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



    return 0;

}



/*

Name: sumOfDigits

Return Type: int

Parameters: int num

Short Description: This function computes the sum of the digits of a given number.

*/
```

```
int sumOfDigits(int num) {

    int sum = 0;



    // Calculate the sum of digits

    while (num != 0) {

        int digit=num%10;

        sum += digit;

        num /= 10;

    }



    return sum;

}
```

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>

#include <stdbool.h> // For boolean data type

bool isPrime(int);

int main() {

    int num;



    // Input the number from the user

    printf("Enter a number: ");

    scanf("%d", &num);



    // Calling the function to check if the number is prime
    if (isPrime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }
    return 0;
}

/*
Name: isPrime
Return Type: bool (boolean)
Parameters: int num
Short Description: This function checks if the given number is prime. It returns true if the number is prime, otherwise false.
*/
bool isPrime(int num) {
    if (num <= 1) {
        return false;
    }

    for (int i = 2; i<=num/2; i++) {
        if (num % i == 0) {
            return false; // If divisible by any number, it is not prime
        }
    }
    return true; // If no factors found, the number is prime

}
```

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 isFibonacci(int);

int main() {

    int num;



    // Input the number from the user

    printf("Enter a number: ");

    scanf("%d", &num);



    // Calling the function to check if the number is part of Fibonacci sequence

    if (isFibonacci(num) == 1) {

        printf("%d is a Fibonacci number.\n", num);

    } else {

        printf("%d is not a Fibonacci number.\n", num);

    }



    return 0;

}



/*
```

```
Name: isFibonacci

Return Type: int

Parameters: int num

Short Description: This function checks if the given number is part of the Fibonacci sequence.

*/

int isFibonacci(int num) {
    int a = 0, b = 1, c;
    // Fibonacci sequence generation until the number exceeds the input
    while (b < num) {
        c = a + b;
        a = b;
        b = c;
    }

    // If the number is in the sequence, return 1 (true), otherwise return 0 (false)
    if (b == num) {
        return 1; // The number is a Fibonacci number
    } else {
        return 0; // The number is not 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.

```
#include <stdio.h>

#include <math.h>
```

```
// Function declaration to calculate the roots of the quadratic equation
void findRoots(int, int, int);

int main() {
    int a, b, c;

    // Input the coefficients from the user
    printf("Enter coefficient a: ");
    scanf("%d", &a);
    printf("Enter coefficient b: ");
    scanf("%d", &b);
    printf("Enter coefficient c: ");
    scanf("%d", &c);

    // Calling the function to find the roots
    findRoots(a, b, c);

    return 0;
}

/*
Name: findRoots
Return Type: void
Parameters: int a, int b, int c
```

```
Short Description: This function calculates the roots of a quadratic equation ax^2 + bx + c = 0 and displays them.

*/

void findRoots(int a, int b, int c) {

    int discriminant = b * b - 4 * a * c;  // Calculate the discriminant



    if (discriminant > 0) {

        // Two real and distinct roots

        double root1 = (-b + sqrt(discriminant)) / (2 * a);

        double root2 = (-b - sqrt(discriminant)) / (2 * a);

        printf("The roots are real and distinct.\n");

        printf("Root 1 = %.2f\n", root1);

        printf("Root 2 = %.2f\n", root2);

    } else if (discriminant == 0) {

        // One real and repeated root

        double root = -b / (2 * a);

        printf("The root is real and repeated.\n");

        printf("Root = %.2f\n", root);

    } else {

        // Complex roots

        double realPart = -b / (2 * a);

        double imaginaryPart = sqrt(-discriminant) / (2 * a);

        printf("The roots are complex.\n");

        printf("Root 1 = %.2f + %.2fi\n", realPart, imaginaryPart);

        printf("Root 2 = %.2f - %.2fi\n", realPart, imaginaryPart);

    }
}
```

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>

// Function declaration to convert binary to decimal

int binaryToDecimal(int);



int main() {

    int binary;



    printf("Enter a binary number: ");

    scanf("%d", &binary);



    // Calling the function to convert binary to decimal

    int decimal = binaryToDecimal(binary);



    // Output the result

    printf("The decimal equivalent is: %d\n", decimal);



    return 0;

}



/*

Name: binaryToDecimal
```

```
Return Type: int

Parameters: int binary

Short Description: This function converts a binary number (passed as an integer) into its decimal equivalent.

*/

int binaryToDecimal(int binary) {

    int decimal = 0, base = 1, remainder;



    while (binary > 0) {

        // Get the last digit (remainder)

        remainder = binary % 10;



        // Add the remainder to the decimal number (multiply by base)

        decimal = decimal + remainder * base;

        // Update the binary number and base

        binary = binary / 10;

        base = base * 2;

    }

    return decimal;

}
```

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>

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

int calculateTrace(int, int, int, int);
int main() {
    int a, b, c, d;

    // Input the elements of the 2x2 matrix from the user
    printf("Enter element a (top-left): ");
    scanf("%d", &a);
    printf("Enter element b (top-right): ");
    scanf("%d", &b);
    printf("Enter element c (bottom-left): ");
    scanf("%d", &c);
    printf("Enter element d (bottom-right): ");
    scanf("%d", &d);

    // Calling the function to compute the trace
    int trace = calculateTrace(a, b, c, d);

    // Output the result
    printf("The trace of the matrix is: %d\n", trace);

    return 0;
}

/*
Name: calculateTrace
Return Type: int
```

```
Parameters: int a, int b, int c, int d

Short Description: This function calculates the trace of a 2x2 matrix (sum of diagonal elements).

*/

int calculateTrace(int a, int b, int c, int d) {

    // Calculate the trace by summing the diagonal elements a and d

    int trace = a + d;

    return trace;

}
```

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>


// Function declaration to check if a number is a palindrome

int isPalindrome(int);


int main() {

    int number;


    // Input the number from the user

    printf("Enter a number: ");

    scanf("%d", &number);


    // Calling the function to check if the number is a palindrome

    int result = isPalindrome(number);


    // Output the result

    if (result == 1) {

        printf("%d is a palindrome.\n", number);
    } else {
        printf("%d is not a palindrome.\n", number);
    }

    return 0;
}

/*
Name: isPalindrome
Return Type: int
Parameters: int number
Short Description: This function checks if the given number is a palindrome and returns 1 if true, else 0.
*/
int isPalindrome(int number) {
    int originalNumber = number;
    int reversedNumber = 0, digit;

    // Reverse the number
    while (number != 0) {
        digit = number % 10;        // Get the last digit
        reversedNumber = reversedNumber * 10 + digit;  // Build the reversed number
        number /= 10;                    // Remove the last digit
    }

    // Check if the original number is equal to the reversed number
    if (originalNumber == reversedNumber) {
        return 1;  // Number is a palindrome
    } else {
        return 0;  // Number 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);

 #include <stdio.h>

// Function declaration

float convert_units(float value, char type);

int main() {

    float value;

    char type;

    printf("Enter the value: \n");

    scanf("%f", &value);

    printf("'C' for cm-to-inches OR 'I' for inches-to-cm\nEnter unit: ");

    scanf(" %c", &type);

    float result=convert_units(value,type);

    switch(type){

        case 'C':

        printf("\nCoverting cm-to-inches=%f\n",result);

        break;

        case 'I':

        printf("Coverting inches-to-cm=%f\n",result);

        break;

        default:

        printf("Invalid conversion type\n");
    }


    return 0;
}


float convert_units(float value, char type){
    float result;
    switch (type){
        case 'C':
        result=value * 0.393701;
        break;
        case 'I':
        result=value * 2.54;
        break;
        default:
        result=-1;
    }
    return result;
}
```

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 totalLength, int pieceLength,int *leftover);
int main() {
    int totalLength, pieceLength, leftover;
    printf("Enter the material length: ");
    scanf("%d", &totalLength);
    printf("Enter the desired length of each piece: ");
    scanf("%d", &pieceLength);

    // Calling the function to calculate the result
    int maxPieces= calculate_cuts(totalLength, pieceLength,&leftover);

    // Output the result
    printf("Maximum number of pieces: %d\n", maxPieces);
    printf("Leftover material: %d units\n", leftover);

    return 0;
}


/*
Name: optimizeCutting
Return Type: void
Parameters: int totalLength, int pieceLength
Short Description: This function calculates the maximum number of pieces and leftover
material based on the given lengths.
*/
int calculate_cuts(int totalLength, int pieceLength,int *leftover) {
    if (pieceLength <= 0) {
        printf("Piece length must be a positive value.\n");
        return 0;
    }

    // Calculate the maximum number of pieces
    int maxPieces = totalLength / pieceLength;

    // Calculate the leftover material
    *leftover = totalLength % pieceLength;

    return maxPieces;
}
```

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);
```

RPM=(Belt Speed/circumference of pulley)×60

```
#include <stdio.h>
#define PI 3.14159
float calculate_rpm(float belt_speed, float pulley_diameter);

int main() {
    float belt_speed, pulley_diameter;
    // Input belt speed and pulley diameter from the user
    printf("Enter the belt speed (m/s): ");
    scanf("%f", &belt_speed);
    printf("Enter the pulley diameter (m): ");
    scanf("%f", &pulley_diameter);

    // Call the function to calculate RPM
    float rpm = calculate_rpm(belt_speed, pulley_diameter);

    // Output the result
    printf("The RPM of the machine is: %.2f\n", rpm);

    return 0;
}
// Function to calculate RPM
float calculate_rpm(float belt_speed, float pulley_diameter) {
    float circumference = PI * pulley_diameter;
    float rpm = (belt_speed / circumference) * 60;  // Calculate RPM (Revolutions Per Minute)
    return rpm;
}
```

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);
```

Production Rate=(Machine Speed×fficiency)/100

```
#include <stdio.h>

int calculate_production_rate(int, int);

int main() {

    int speed,efficiency;

    // Input belt speed and pulley diameter from the user

    printf("Enter the machine speed (units per hour): ");

    scanf("%d", &speed);

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

    scanf("%d", &efficiency);

    // Call the function to calculate RPM

    int rate=calculate_production_rate(speed,efficiency);

    // Output the result

    printf("The effective production rate : %d\n", rate);

    return 0;

}



// Function to calculate rate

int calculate_production_rate(int speed, int efficiency){

    int rate=(speed*efficiency)/100;

    return rate;

}
```

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);



int main() {

    int total_length, leftover_length;


    // Input the total material length and leftover material length

    printf("Enter the total material length: ");

    scanf("%d", &total_length);

    printf("Enter the leftover material length: ");

    scanf("%d", &leftover_length);

    // Call the function to calculate wastage

    int wastage = calculate_wastage(total_length, leftover_length);

    // Output the result

    printf("The material wasted is: %d units\n", wastage);

    return 0;

}


int calculate_wastage(int total_length, int leftover_length) {

    return total_length - leftover_length;

}
```

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);
```

Total Energy Cost=Power Rating (kW)×Operating Hours×Cost per kWh

```
#include <stdio.h>
float calculate_energy_cost(float power_rating, float hours, float cost_per_kwh);
int main() {
    float power_rating, hours, cost_per_kwh;
    // Input power rating, operating hours, and cost per kWh from the user
    printf("Enter the power rating of the device (kW): ");
    scanf("%f", &power_rating);
    printf("Enter the operating hours: ");
    scanf("%f", &hours);
    printf("Enter the cost per kWh: ");
    scanf("%f", &cost_per_kwh);

    // Call the function to calculate the energy cost
    float total_cost = calculate_energy_cost(power_rating, hours, cost_per_kwh);

    // Output the result
    printf("The total energy cost is: %.2f\n", total_cost);
    return 0;
}

// Function to calculate energy cost
float calculate_energy_cost(float power_rating, float hours, float cost_per_kwh) {
    return power_rating * hours * cost_per_kwh;
}
```

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);
```

Heat Generated (J)=Power Usage (W)×(1−(Efficiency (%)/100))×Time (s), Here for 1 s

```
#include <stdio.h>

float calculate_heat(float power_usage, float efficiency);

int main() {

    float power_usage, efficiency;

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

    scanf("%f", &power_usage);

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

    scanf("%f", &efficiency);

    // Call the function to calculate heat generated

    float heat_generated = calculate_heat(power_usage, efficiency);

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

    return 0;

}

// Function to calculate heat generation

float calculate_heat(float power_usage, float efficiency) {

    return power_usage * (1 - (efficiency / 100));

}
```

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);

 float wear_rate = time * wear_rate_per_hour;

#include <stdio.h>

float calculate_wear_rate(float time, int material_type);

int main() {
    float time;
    int material_type;


    printf("Enter the operating time (hours): ");
    scanf("%f", &time);
    printf("Enter the material type (1 for Metal, 2 for Plastic): ");
    scanf("%d", &material_type);
    // Call the function to calculate wear rate
    float wear_rate = calculate_wear_rate(time, material_type);
    if (wear_rate != -1) { // Check for valid wear rate
        printf("The wear rate is: %.2f%%\n", wear_rate);
    }
    return 0;
}



float calculate_wear_rate(float time, int material_type) {
    float wear_rate_per_hour;
    // Determine wear rate per hour based on material type
    if (material_type == 1) {
        wear_rate_per_hour = 2.0; // 2% per hour for metal
    } else if (material_type == 2) {
        wear_rate_per_hour = 1.0; // 1% per hour for plastic
    } else {
        printf("Invalid material type!\n");
        return -1;
    }
    // Calculate the total wear rate
    float wear_rate = time * wear_rate_per_hour;
    return wear_rate;
}
```

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);
```

Reorder Quantity=Consumption Rate×Lead Time

```
#include<stdio.h>
int calculate_reorder_quantity(int consumption_rate, int lead_time);


int main(){
    int consumptionRate,leadTime;
    printf("Enter the consumption rate:\n");
    scanf("%d",&consumptionRate);
     printf("Enter the lead time:\n");
    scanf("%d",&leadTime);
    int Quantity=calculate_reorder_quantity(consumptionRate,leadTime);
    printf("The recoder quantity =%d",Quantity);


    return 0;
}
int calculate_reorder_quantity(int rate,int time){
    return rate*time;
}
```

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);
```

Defective Rate (%)=(Defective Items/Batch size)×100

```
#include <stdio.h>
float calculate_defective_rate(int defective_items, int batch_size);


int main() {
    int defective_items, batch_size;
    printf("Enter the number of defective items: ");
    scanf("%d", &defective_items);
    printf("Enter the total batch size: ");
    scanf("%d", &batch_size);


    float defective_rate = calculate_defective_rate(defective_items, batch_size);
    if (defective_rate != -1) {
        printf("The defective rate is: %.2f%%\n", defective_rate);
    }

    return 0;
}
float calculate_defective_rate(int defective_items, int batch_size) {
    if (batch_size == 0) {
        printf("Error: Batch size cannot be zero.\n");
        return -1; // Return an error value
    }
    return ((float)defective_items / batch_size) * 100;
}
```

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);
```

Efficiency (%)=(Effective Working Time/Total time)×100

```
#include <stdio.h>
float calculate_efficiency(int output_rate, int downtime);

int main() {
    int output_rate, downtime;
    printf("Enter the output rate (units/hour): ");
    scanf("%d", &output_rate);
    printf("Enter the downtime (minutes): ");
    scanf("%d", &downtime);
    float efficiency = calculate_efficiency(output_rate, downtime);
    if (efficiency != -1) { // Check for valid result

        printf("The assembly line efficiency is: %.2f%%\n", efficiency);
    }

    return 0;
}
float calculate_efficiency(int output_rate, int downtime) {
    const int total_time_minutes = 60; // Total time in minutes for an hour
    int effective_time = total_time_minutes - downtime;

    if (effective_time < 0) {
        printf("Error: Downtime cannot exceed total time.\n");
        return -1; // Return an error value
    }

    // Efficiency calculation
    return ((float)effective_time / total_time_minutes) * 100;
}
```

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);
```

Required Paint (liters)=Surface Area (m^2)/Paint coverage

```
#include <stdio.h>

float calculate_paint(float area, float coverage);



int main() {
    float surface_area, paint_coverage;
    printf("Enter the surface area to be painted (m²): ");
    scanf("%f", &surface_area);
    printf("Enter the paint coverage per liter (m²/liter): ");
    scanf("%f", &paint_coverage);

    float required_paint = calculate_paint(surface_area, paint_coverage);

    if (required_paint != -1) { // Check for valid result
        // Output the result
        printf("The required paint is: %.2f liters\n", required_paint);
    }

    return 0;
}
```

```
float calculate_paint(float area, float coverage) {
    if (coverage <= 0) {
        printf("Error: Paint coverage must be greater than 0.\n");
        return -1;
    }
    return area / coverage;
}
```

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);
```

Hours Remaining=Maintenance Interval−(Current Usage % Maintenance Interval)

```
#include <stdio.h>

int calculate_maintenance_schedule(int current_usage, int interval);

int main() {
    int current_usage, interval;
    printf("Enter the current machine usage (hours): ");
    scanf("%d", &current_usage);
    printf("Enter the maintenance interval (hours): ");
    scanf("%d", &interval);
    int remaining_hours = calculate_maintenance_schedule(current_usage, interval);


    if (remaining_hours != -1) { // Check for valid result
        printf("Hours remaining for maintenance: %d\n", remaining_hours);
    }
    return 0;
}
int calculate_maintenance_schedule(int current_usage, int interval) {
    if (interval <= 0) {
        printf("Error: Maintenance interval must be greater than 0.\n");
        return -1;
    }

    int remaining_hours = interval - (current_usage % interval);
    return remaining_hours;
}
```

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);
```

Cycle Time=3600/(Machine Speed×Number of Operations per Cycle)

```
#include <stdio.h>
float calculate_cycle_time(int speed, int operations);
int main() {
    int speed, operations;
    printf("Enter the machine speed (units per hour): ");
    scanf("%d", &speed);
    printf("Enter the number of operations per cycle: ");
    scanf("%d", &operations);
    float cycle_time = calculate_cycle_time(speed, operations);
    if (cycle_time != -1.0) {
        printf("Optimal cycle time: %.2f seconds\n", cycle_time);
    }
    return 0;
}


float calculate_cycle_time(int speed, int operations) {
    if (speed <= 0 || operations <= 0) {
        printf("Error: Speed and operations must be greater than zero.\n");
        return -1;
    }
    float cycle_time = 3600.0 / (speed * operations);
    return cycle_time;
}
```

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 price of item:\n");

    scanf("%f",&originalPrice);

    printf("Enter Discount percentage\n");

    scanf("%f",&discountPercentage);
    calculateDiscount(originalPrice,discountPercentage);



    return 0;

}



void calculateDiscount(float price, float discount){

    float discountAmount=(price*discount)/100;

    printf("Original price =%f\n",price);

    printf("Discounted price =%f\n",discountAmount);

}
```

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 currentInventory, quantityChange;

    printf("Enter the current inventory count: ");

    scanf("%d", &currentInventory);



    printf("Enter the quantity to add/remove (positive to add, negative to remove): ");

    scanf("%d", &quantityChange);



    int updatedInventory = updateInventory(currentInventory, quantityChange);
    printf("The updated inventory count is: %d\n", updatedInventory);

    return 0;
}

int updateInventory(int currentCount, int changeQuantity) {
    int newCount = currentCount + changeQuantity;
    return newCount;
}
```

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 price, taxRate;
    printf("Enter the price of the item: ");
    scanf("%f", &price);
    printf("Enter the sales tax rate (in percentage): ");
    scanf("%f", &taxRate);

    float total = calculateTotalPrice(price, taxRate);
    printf("The total price after tax is: %.2f\n", total);

    return 0;
}
float calculateTotalPrice(float itemPrice, float taxRate) {

    float totalPrice = itemPrice + (itemPrice * taxRate / 100);
    return totalPrice;
}
```

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

    return 0;
}
int calculateLoyaltyPoints(float amountSpent) {
    int points = (int)(amountSpent / 10);
    return points;
}
```

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;
    printf("Enter the number of items\n");
    scanf("%d",&numberOfItems);
    float prices[numberOfItems];
    for(int i=0;i<numberOfItems;i++){
        printf("Prices[%d]\n",i+1);
        scanf("%f",&prices[i]);
    }
    //The correct way to pass the array to the function is without the [] inside the function call
    int total_price=calculateOrderTotal(prices,numberOfItems);
    printf("Total price=%d\n",total_price);
    return 0;


}
float calculateOrderTotal(float prices[], int numberOfItems){
    float total=0;
    //each item has a different price, you need to sum the prices of all the items individually, rather than multiplying the number of items by a single price.
    for(int i=0;i<numberOfItems;i++){
        total+=prices[i];
    }
    return total;
}
```

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;
    printf("Enter the item's price: ");
    scanf("%f", &itemPrice);
    printf("Enter the refund percentage: ");
    scanf("%f", &refundPercentage);
    float refundAmount = calculateRefund(itemPrice, refundPercentage);
    printf("The refund amount is: %.2f\n", refundAmount);
    return 0;
}


float calculateRefund(float itemPrice, float refundPercentage) {
    return (itemPrice * refundPercentage) / 100.0;
}
```

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 weight;
    printf("Enter the weight of the package (in kg): ");
    scanf("%f", &weight);
    float shippingCost = calculateShippingCost(weight);
    if(shippingCost!=-1)
    printf("The shipping cost is: $%.2f\n", shippingCost);

    return 0;
}
float calculateShippingCost(float weight) {
    float cost;

    if (weight <= 5) {
        cost = 5.0; // $5 for up to 5kg
    } else if (weight <= 10) {
        cost = 10.0; // $10 for 5-10kg
    } else {
        printf("Cant access this weight\n");
        return -1;
       //cost = 10.0 + (weight - 10) * 2.0; // $2 for each additional kg over 10kg
    }


    return cost;
}
```

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;
    printf("Enter the amount in the original currency: ");
    scanf("%f", &amount);


    printf("Enter the exchange rate (target currency per unit of original currency): ");
    scanf("%f", &exchangeRate);
    float convertedAmount = convertCurrency(amount, exchangeRate);
    printf("The converted amount is: %.2f (in the target currency)\n", convertedAmount);
    return 0;
}
float convertCurrency(float amount, float exchangeRate) {
    return amount * exchangeRate;
}
```

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;

    printf("Enter the price from Vendor A: ");

    scanf("%f", &priceA);

    printf("Enter the price from Vendor B: ");

    scanf("%f", &priceB);

    float lowerPrice = findLowerPrice(priceA, priceB);

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



    return 0;

}

float findLowerPrice(float priceA, float priceB) {

    return (priceA < priceB) ? priceA : priceB; // Return the lower of the two prices

}
```

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);
    // Check eligibility for senior citizen 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;
}

bool isEligibleForSeniorDiscount(int age) {
    return age >= 60; // Eligibility threshold is 60 years or older
}
```