```
//WAP to implement a function which is going to add two number

#include <stdio.h>
void sum2Elements(int, int);

int main(){
    int a = 20, b = 30;
    //call by value
    sum2Elements(a, b);
    printf("001 a=%d and b=%d",a,b);

   return 0;
}

/*
Name: fun()
Return Type: void
Parameter:(data type of each parameter): No parameters
Shord disciption: it is used to tract the number of times the
            function is getting called
*/
//function definition
void sum2Elements(int d,int e){
   e=30;
   d=40;
   printf("002=%d and e=%d \n",d,e);
   int sum = 0;
   sum = d + e;
   printf("Sum = %d \n",sum);
}
```

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 , int , int);

int main()
{
   int a =4 , b=3;
   int temp;
   swap( a, b, temp);
   printf("001 a=%d and b= %d",a,b);

   return 0;
}
void swap(int a, int b, int temp)
{
    a= 4,b=3;
    temp=a;
    a=b;
    b=temp;
    printf("a=%d and b = %d\n",a,b);

}
```

WITH RETURN

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

```
// Function to swap two numbers and return the swapped values
void swap(int a, int b, int *new_a, int *new_b) {
   int temp = a;
   *new_a = b;
   *new_b = temp;
}

int main() {
   int a = 4, b = 3;
   int new_a, new_b;

   swap(a, b, &new_a, &new_b);
   printf("Original a=%d and b=%d\n", a, b);
   printf("Swapped a=%d and b=%d\n", new_a, new_b);

   return 0;
}
```

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>
void max(int, int);

int main()
{
   int a=4,b=5;

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

```
    return 0;
}
void max(int a, int b){

    a=5,b=6;
    if(a>b){
        printf("%d is larger\n",a);
    }else {
         printf("%d is larger\n",b);
    }
}
```

WITH RETURN TYPE

```
#include<stdio.h>

// Function to return the larger of two numbers
int max(int a, int b) {
    return (a > b) ? a : b;
}

int main() {
    int a = 4, b = 5;
    int larger = max(a, b);  // Call the max function and store the result

    printf("The larger number is %d\n", larger);
    printf("a = %d and b = %d\n", a, b);

    return 0;
}
```

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>
void fact(int);

int main()
{
   int a=4;

   fact(a);
   printf("a=%d",a);

   return 0;
}
void fact(int a){
   a=5;
   int fact=1;
   for(int i=1;i<=a;i++){
      fact=fact*i;

   } printf("factorial of %d=%d\n",a,fact);
}
```

WITH RETURN

```
#include <stdio.h>

// Function to calculate the factorial of a number and return
int fact(int a) {
   int fact = 1;
   for (int i = 1; i <= a; i++) {
      fact *= i;
   }
   return fact;
}

int main() {
   int a = 4;
   int factorial = fact(a);  // Call the fact function and store the result

   printf("Factorial of %d = %d\n", a, factorial);
   printf("a = %d\n", a);

   return 0;
}
```

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 checkEven_odd(int,int);
```

```
int main()
{
   int a=5 , b=6;
   checkEven_odd(a,b);
   printf(" a= %d and b =%d",a,b);

   return 0;
}
void checkEven_odd(int a,int b){
   a=4;
   b=5;
   if(a%2==0){
      printf("%d is 0dd\n",a);
   }
   if(b%2==0){
      printf("%d is 0dd\n",b);
   }

}
```

WITHOUT RETURN

```
#include <stdio.h>

// Function to check if the number is even or odd and return the result
// Returns 1 for even, 0 for odd
int checkEvenOdd(int num) {
   if (num % 2 == 0) {
      return 1; // Even
   } else {
```

```
        return 0; // Odd
    }
}

int main() {
    int number;
    int result;

    // Ask user to input a number
    printf("Enter an integer: ");
    scanf("%d", &number);

    // Call the function to check if the number is even or odd
    result = checkEvenOdd(number);

    // Output the result
    if (result == 1) {
        printf("%d is even.\n", number);
    } else {
        printf("%d is odd.\n", number);
    }

    return 0;
}
```

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 calculateSimpleInterest(float, float, float);

int main() {
    float principal, rate, time, interest;


    printf("Enter the principal amount: ");
    scanf("%f", &principal);
    printf("Enter the rate of interest: ");
    scanf("%f", &rate);
    printf("Enter the time period in years: ");
    scanf("%f", &time);


    interest = calculateSimpleInterest(principal, rate, time);


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

    return 0;
}
float calculateSimpleInterest(float principal, float rate, float time) {
    return (principal * rate * time) / 100;
}
```

WITH RETURN

```
#include <stdio.h>


float calculateSimpleInterest(float principal, float rate, float time) {
    return (principal * rate * time) / 100;
}

int main() {
    float principal, rate, time, interest;

    // Input principal, rate, and time from the user
    printf("Enter the principal amount: ");
    scanf("%f", &principal);
    printf("Enter the rate of interest: ");
    scanf("%f", &rate);
    printf("Enter the time period in years: ");
    scanf("%f", &time);

    // Calculate simple interest
    interest = calculateSimpleInterest(principal, rate, time);

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

    return 0;
}
```

Reverse a Number

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

```
/*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>


// Function to reverse a number



int main() {
   int number;


   printf("Enter an integer: ");
   scanf("%d", &number);


   printf("Original number before function call: %d\n", number);


   return 0;
}
void reverseNumber(int num) {
   int reversed = 0;
   while (num != 0) {
      int digit = num % 10;
      reversed = reversed * 10 + digit;
      num /= 10;
```

```
    }

}
```

WITHOUT RETURN

```
#include <stdio.h>

// Function to reverse a number


int main() {
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    printf("Original number before function call: %d\n", number);

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

```
}
```

GCD of Two Numbers

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

```
#include <stdio.h>

int main() {
    int num1, num2;

    printf("Enter the first number: ");
    scanf("%d", &num1);

    printf("Enter the second number: ");
    scanf("%d", &num2);

    int result = gcd(num1, num2);

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

    return 0;
}
// Function to calculate the GCD of two numbers using the Euclidean algorithm
```

```
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

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 number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    int result = sumOfDigits(number);

    printf("The sum of the digits of %d is %d.\n", number, result);

    return 0;
}
```

```
// Function to compute the sum of the digits of a number
int sumOfDigits(int num) {
   int sum = 0;
   while (num != 0) {
      sum += num % 10;
      num /= 10;
   }
   return sum;
}
```

WITHOUT RETURN

```
#include <stdio.h>

void sumOfDigits(int);
int main() {
   int number;

   printf("Enter an integer: ");
   scanf("%d", &number);

   sumOfDigits(number);

   return 0;
}

// Function to compute the sum of the digits of a number
void sumOfDigits( num)
{
   int sum = 0;
```

```
    while (num != 0) {
        sum += num % 10;
        num /= 10;

    }
    printf("sum of numbers is %d",sum);

}
```

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 isPrime(int);

int main() {
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (isPrime(number)) {
        printf("%d is a prime number.\n", number);
    } else {
        printf("%d is not a prime number.\n", number);
    }
```

```
    return 0;
}

// Function to check if a number is prime
int isPrime(int num) {
    if (num <= 1) {
        return 0; // false
    }
    if (num == 2) {
        return 1; // true
    }
    if (num % 2 == 0) {
        return 0; // false
    }
    for (int i = 3; i * i <= num; i += 2) {
        if (num % i == 0) {
            return 0; // false
        }
    }
    return 1; // true
}
```

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>
#include <math.h>

// Function to check if a number is a perfect square
int is_perfect_square(int n) {
```

```
    int sqrt_n = (int)sqrt(n);
    return (sqrt_n * sqrt_n == n);
}

// Function to check if a number is in the Fibonacci sequence
int is_fibonacci(int num) {
    // Store the value of the input number
    int value = num;

    // Check the two conditions for Fibonacci numbers
    if (is_perfect_square(5 * value * value + 4) || is_perfect_square(5 * value * value - 4)) {
        return value;  // Return the stored value if the number is a Fibonacci number
    } else {
        return -1;  // Return -1 if the number is not a Fibonacci number
    }
}

int main() {
    int num;
    printf("Enter a number to check if it's a Fibonacci number: ");
    scanf("%d", &num);

    int result = is_fibonacci(num);
    if (result != -1) {
        printf("%d is a Fibonacci number.\n", result);
    } else {
        printf("%d is NOT a Fibonacci number.\n", num);
    }

    return 0;
}
```

WITHOUT RETURN

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

// Function to check if a number is a perfect square
int is_perfect_square(int n) {
    int sqrt_n = (int)sqrt(n);
    return (sqrt_n * sqrt_n == n);
}

// Function to check if a number is in the Fibonacci sequence
void is_fibonacci(int num) {
    // Check the two conditions for Fibonacci numbers
    if (is_perfect_square(5 * num * num + 4) || is_perfect_square(5 * num * num - 4)) {
        printf("%d is a Fibonacci number.\n", num);  // Print if it's a Fibonacci number
    } else {
        printf("%d is NOT a Fibonacci number.\n", num);  // Print if it's not a Fibonacci number
    }
}

int main() {
    int num;
    printf("Enter a number to check if it's a Fibonacci number: ");
    scanf("%d", &num);

    is_fibonacci(num);  // Call the function to check and print the result

    return 0;
}
```

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>    // For sqrt() function

// Function to calculate roots of the quadratic equation and store the values in the provided pointers
int calculate_roots(double a, double b, double c, double *root1_real, double *root1_imag, double *root2_real, double *root2_imag) {
    double discriminant = b * b - 4 * a * c;
    double realPart = -b / (2 * a);

    // Case 1: Two real roots
    if (discriminant > 0) {
        *root1_real = realPart + sqrt(discriminant) / (2 * a);
        *root2_real = realPart - sqrt(discriminant) / (2 * a);
        *root1_imag = *root2_imag = 0; // No imaginary part for real roots
        return 1; // Two real roots
    }
    // Case 2: One real root (repeated root)
    else if (discriminant == 0) {
        *root1_real = *root2_real = realPart;
        *root1_imag = *root2_imag = 0; // No imaginary part for repeated real root
        return 0; // One real root (repeated)
    }
    // Case 3: Two complex roots
    else {
        *root1_real = *root2_real = realPart;
        *root1_imag = sqrt(-discriminant) / (2 * a);
```

```
        *root2_imag = -(*root1_imag); // Complex conjugates
        return -1; // Two complex roots
    }
}

int main() {
    double a, b, c;
    double root1_real, root1_imag, root2_real, root2_imag;

    // Taking input for coefficients
    printf("Enter coefficients a, b, and c: ");
    scanf("%lf %lf %lf", &a, &b, &c);

    // Call the function to calculate and store roots
    int result = calculate_roots(a, b, c, &root1_real, &root1_imag, &root2_real, &root2_imag);

    // Display the roots
    if (result == 1) {
        printf("Root 1: %.2lf\n", root1_real);
        printf("Root 2: %.2lf\n", root2_real);
    } else if (result == 0) {
        printf("Root: %.2lf\n", root1_real); // Only one root for repeated real root
    } else {
        printf("Root 1: %.2lf + %.2lfi\n", root1_real, root1_imag);
        printf("Root 2: %.2lf - %.2lfi\n", root2_real, root2_imag);
    }

    return 0;
}
```

WITHOUT RETURN

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

void calculate_roots(double a, double b, double c, double *root1_real, double *root1_imag, double *root2_real, double *root2_imag) {
    double discriminant = b * b - 4 * a * c;
    double realPart = -b / (2 * a);

    // Case 1: Two real roots
    if (discriminant > 0) {
        *root1_real = realPart + sqrt(discriminant) / (2 * a);
        *root2_real = realPart - sqrt(discriminant) / (2 * a);
        *root1_imag = *root2_imag = 0; // No imaginary part for real roots
    }
    // Case 2: One real root (repeated root)
    else if (discriminant == 0) {
        *root1_real = *root2_real = realPart;
        *root1_imag = *root2_imag = 0; // No imaginary part for repeated real root
    }
    // Case 3: Two complex roots
    else {
        *root1_real = *root2_real = realPart;
        *root1_imag = sqrt(-discriminant) / (2 * a);
        *root2_imag = -(*root1_imag); // Complex conjugates
    }
}
```

```
int main() {
    double a, b, c;
    double root1_real, root1_imag, root2_real, root2_imag;

    // Taking input for coefficients
    printf("Enter coefficients a, b, and c: ");
    scanf("%lf %lf %lf", &a, &b, &c);

    // Call the function to calculate and store roots
    calculate_roots(a, b, c, &root1_real, &root1_imag, &root2_real, &root2_imag);

    // Display the roots
    if (root1_imag == 0 && root2_imag == 0) {
        printf("Root 1: %.2lf\n", root1_real);
        printf("Root 2: %.2lf\n", root2_real);
    } else {
        printf("Root 1: %.2lf + %.2lfi\n", root1_real, root1_imag);
        printf("Root 2: %.2lf - %.2lfi\n", root2_real, root2_imag);
    }

    return 0;
}
```

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 to convert a binary number to its decimal equivalent
int binaryToDecimal(int binary) {
    int decimal = 0, base = 1, remainder;

    while (binary > 0) {
        remainder = binary % 10;
        decimal = decimal + remainder * base;
        binary = binary / 10;
        base = base * 2;
    }
    return decimal;
}

int main() {
    int binaryNumber;

    printf("Enter a binary number: ");
    scanf("%d", &binaryNumber);

    int decimalNumber = binaryToDecimal(binaryNumber);

    printf("The decimal equivalent of binary number %d is %d.\n", binaryNumber, decimalNumber);

    return 0;
}
```

WITHOUT RETURN

```
#include <stdio.h>

// Function to convert a binary number to its decimal equivalent
```

```
void binaryToDecimal(int binary, int* decimal) {
    *decimal = 0;
    int base = 1, remainder;

    while (binary > 0) {
        remainder = binary % 10;
        *decimal = *decimal + remainder * base;
        binary = binary / 10;
        base = base * 2;
    }
}

int main() {
    int binaryNumber, decimalNumber;

    printf("Enter a binary number: ");
    scanf("%d", &binaryNumber);

    binaryToDecimal(binaryNumber, &decimalNumber);

    printf("The decimal equivalent of binary number %d is %d.\n", binaryNumber, decimalNumber);

    return 0;
}
```

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 to compute the trace of a 2x2 matrix
int computeTrace(int a11, int a12, int a21, int a22) {
    return a11 + a22; // Sum of the diagonal elements
}

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

    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);

    int trace = computeTrace(a11, a12, a21, a22);

    printf("The trace of the 2x2 matrix is: %d\n", trace);

    return 0;
}
```

WITHOUT RETURN

```
#include <stdio.h>

// Function to compute the trace of a 2x2 matrix
void computeTrace(int a11, int a12, int a21, int a22, int *trace) {
```

```
    *trace = a11 + a22; // Sum of the diagonal elements
}

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

    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);

    computeTrace(a11, a12, a21, a22, &trace);

    printf("The trace of the 2x2 matrix is: %d\n", trace);

    return 0;
}
```

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 to check if a number is a palindrome
int isPalindrome(int num) {
    int originalNum = num;
    int reversedNum = 0;

    // Reversing the number
    while (num != 0) {
        int digit = num % 10;
        reversedNum = reversedNum * 10 + digit;
        num /= 10;
    }

    // Check if the original number and the reversed number are the same
    return (originalNum == reversedNum);
}

int main() {
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

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

    return 0;
}
```

WITHOUT RETURN

```
#include <stdio.h>

// Function to check if a number is a palindrome
void isPalindrome(int num, int *result) {
   int originalNum = num;
   int reversedNum = 0;

   // Reversing the number
   while (num != 0) {
      int digit = num % 10;
      reversedNum = reversedNum * 10 + digit;
      num /= 10;
   }

   // Check if the original number and the reversed number are the same
   *result = (originalNum == reversedNum);
}

int main() {
   int number;
   int isPalin;

   printf("Enter an integer: ");
   scanf("%d", &number);

   isPalindrome(number, &isPalin);

   if (isPalin) {
```

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

    return 0;
}
```

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>
float convert_units(float value, char type);

int main() {
    float value;
    char type;
    printf("Enter the measurement value: ");
    scanf("%f", &value);
    printf("Enter the conversion type (C for cm-to-inches, I for inches-to-cm): ");
    scanf(" %c", &type);

    // Perform the conversion
    float converted_value = convert_units(value, type);
```

```
    // Check for valid conversion and print the result
    if (converted_value != -1) {
        printf("The converted value is: %.2f\n", converted_value);
    }

    return 0;
}

float convert_units(float value, char type) {
    if (type == 'C'|| type =='c') {
        // Convert from cm to inches
        return value / 2.54;
    } else if (type == 'I'|| type=='i') {
        // Convert from inches to cm
        return value * 2.54;
    } else {
        // Invalid conversion type
        printf("Invalid conversion type. Please use 'C' for cm-to-inches or 'I' for inches-to-cm.\n");
        return -1;
    }
}
```

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() {
    int material_length, piece_length;
    int num_pieces, leftover;

    printf("Enter the total length of the raw material: ");
    scanf("%d", &material_length);
    printf("Enter the desired length of each piece: ");
    scanf("%d", &piece_length);

    // Calculate the number of pieces
    num_pieces = calculate_cuts(material_length, piece_length);

    // Calculate the leftover material
    leftover = material_length % piece_length;
    printf("Maximum number of pieces that can be cut: %d\n", num_pieces);
    printf("Leftover material: %d\n", leftover);

    return 0;
}
```

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>

// Function to calculate RPM
float calculate_rpm(float belt_speed, float pulley_diameter) {
    return (belt_speed * 60) / (pulley_diameter * 3.14);
}

int main() {
    float belt_speed, pulley_diameter, rpm;

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

    // Calculate RPM
    rpm = calculate_rpm(belt_speed, pulley_diameter);

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

    return 0;
}
```

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);
```

```c
#include <stdio.h>

int calculate_production_rate(int speed, int efficiency);

int main() {
    int speed, efficiency, production_rate;

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

    // Calculate the effective production rate
    production_rate = calculate_production_rate(speed, efficiency);

    // Print the result
    printf("The effective production rate is: %d units per hour\n", production_rate);

    return 0;
}

// Function to calculate the effective production rate
int calculate_production_rate(int speed, int efficiency) {
    return (speed * efficiency) / 100;
}
```

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;
   int wastage;
   printf("enter the total lenght of the material :");
   scanf("%d",&total_length);
   printf("enter the length pf the material required :");
   scanf("%d",&leftover_length);

   wastage = calculate_wastage(total_length,leftover_length);

   printf("leftover material length =%d",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);
```

```
#include <stdio.h>

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

int main() {
    float power_rating, hours, cost_per_kwh, total_cost;

    printf("Enter the power rating (kW): ");
    scanf("%f", &power_rating);
    printf("Enter the operating hours: ");
    scanf("%f", &hours);
    printf("Enter the cost per kWh: ");
    scanf("%f", &cost_per_kwh);

    // Calculate the total energy cost
    total_cost = calculate_energy_cost(power_rating, hours, cost_per_kwh);

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

    return 0;
```

}

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>

int main() {

    float power_usage, efficiency, heat_generated;
    printf("Enter the power usage (Watts): ");
    scanf("%f", &power_usage);
    printf("Enter the efficiency (%%): ");
    scanf("%f", &efficiency);

    // Calculating the heat generated
    heat_generated = calculate_heat(power_usage, efficiency);
    printf("The heat generated is: %.2f Joules\n", heat_generated);

    return 0;
}

float calculate_heat(float power_usage, float efficiency) {
    float power_loss = power_usage * (1 - efficiency / 100);
    return power_loss * 3600; // Convert power loss (Watts) to heat (Joules) over one hour
}
```

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;

    switch (material_type) {
        case 1: // Metal
            wear_rate = time * 0.5; // Example rate for metal
            break;
        case 2: // Plastic
            wear_rate = time * 0.3; // Example rate for plastic
            break;
        default:
            printf("Invalid material type. Please use 1 for metal or 2 for plastic.\n");
            wear_rate = -1;
            break;
    }

    return wear_rate;
}

int main() {
    float time;
    int material_type;
    float wear_rate;
```

```
    printf("Enter the operating time (hours): ");
    scanf("%f", &time);
    printf("Enter the material type (1 for metal, 2 for plastic): ");
    scanf("%d", &material_type);

    // Calculate the wear rate
    wear_rate = calculate_wear_rate(time, material_type);

    // Check for valid wear rate
    if (wear_rate != -1) {
        printf("The wear rate is: %.2f%%\n", wear_rate);
    }

    return 0;
}
```

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 to calculate reorder quantity
int calculate_reorder_quantity(int consumption_rate, int lead_time) {
    return consumption_rate * lead_time;
}
```

```
int main() {
    int consumption_rate, lead_time, reorder_quantity;

    printf("Enter the consumption rate (units/day): ");
    scanf("%d", &consumption_rate);
    printf("Enter the lead time (days): ");
    scanf("%d", &lead_time);

    // Calculate the reorder quantity
    reorder_quantity = calculate_reorder_quantity(consumption_rate, lead_time);

    printf("The reorder quantity is: %d units\n", reorder_quantity);

    return 0;
}
```

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>

// Function to calculate defective rate
float calculate_defective_rate(int defective_items, int batch_size) {
    return ((float)defective_items / batch_size) * 100;
}

int main() {
    int defective_items, batch_size;
```

```
    float defective_rate;

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

    // Calculate the defective rate
    defective_rate = calculate_defective_rate(defective_items, batch_size);

    // Print the result
    printf("The defective rate is: %.2f%%\n", defective_rate);

    return 0;
}
```

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>

// Function to calculate efficiency
float calculate_efficiency(int output_rate, int downtime) {
    int total_time = 60; // Total time in minutes for one hour
    float operational_time = total_time - downtime;
    return (operational_time / total_time) * 100; // Efficiency as a percentage
}
```

```
int main() {
    int output_rate, downtime;
    float efficiency;

    printf("Enter the output rate (units/hour): ");
    scanf("%d", &output_rate);
    printf("Enter the downtime (minutes): ");
    scanf("%d", &downtime);

    // Calculate the efficiency
    efficiency = calculate_efficiency(output_rate, downtime);
    printf("The efficiency of the assembly line is: %.2f%%\n", efficiency);

    return 0;
}
```

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>

// Function to calculate required paint
float calculate_paint(float area, float coverage) {
    return area / coverage;
}
```

```
int main() {
    float area, coverage, required_paint;

    printf("Enter the surface area (m²): ");
    scanf("%f", &area);
    printf("Enter the paint coverage per liter (m²/liter): ");
    scanf("%f", &coverage);

    // Calculate the required paint
    required_paint = calculate_paint(area, coverage);
    printf("The required paint is: %.2f liters\n", required_paint);

    return 0;
}
```

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>

// Function to calculate hours remaining for maintenance
int calculate_maintenance_schedule(int current_usage, int interval) {
    return interval - (current_usage % interval);
}

int main() {
    int current_usage, interval, hours_remaining;
```

```
    printf("Enter the current usage (hours): ");
    scanf("%d", &current_usage);
    printf("Enter the maintenance interval (hours): ");
    scanf("%d", &interval);



    hours_remaining = calculate_maintenance_schedule(current_usage, interval);

    // Print the result
    printf("Hours remaining for maintenance: %d\n", hours_remaining);

    return 0;
}
```

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>

// Function to calculate optimal cycle time
float calculate_cycle_time(int speed, int operations) {
    // Convert machine speed from units per hour to units per second
    float speed_per_second = speed / 3600.0;
    // Calculate cycle time in seconds
    return operations / speed_per_second;
}
```

```
int main() {
   int speed, operations;
   float cycle_time;



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

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

   return 0;
}
```

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, float *discountedPrice);

void calculateDiscount(float originalPrice, float discountPercentage, float *discountedPrice) {
```

```
    *discountedPrice = originalPrice - (originalPrice * discountPercentage / 100);
}

int main() {
    float originalPrice, discountPercentage, discountedPrice;

    // Input the original price and discount percentage
    printf("Enter the original price: ");
    scanf("%f", &originalPrice);
    printf("Enter the discount percentage: ");
    scanf("%f", &discountPercentage);

    // Calculate the discounted price
    calculateDiscount(originalPrice, discountPercentage, &discountedPrice);

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

    return 0;
}
```

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) {
    return currentCount + changeQuantity;
```

```
}

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

    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);

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

    return 0;
}
```

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>

// Function to calculate the total price
float calculateTotalPrice(float itemPrice, float taxRate) {
    return itemPrice + (itemPrice * taxRate / 100);
}

int main() {
```

```
    float itemPrice, taxRate, totalPrice;

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

    // Calculate the total price after tax
    totalPrice = calculateTotalPrice(itemPrice, taxRate);

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

    return 0;
}
```

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>

// Function to calculate loyalty points
int calculateLoyaltyPoints(float amountSpent, float conversionRate) {
    return (int)(amountSpent / conversionRate);
}

int main() {
    float amountSpent, conversionRate;
```

```
    int loyaltyPoints;

    printf("Enter the amount spent: ");
    scanf("%f", &amountSpent);
    printf("Enter the conversion rate (e.g., 10 for 1 point per $10 spent): ");
    scanf("%f", &conversionRate);

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

    // Print the result
    printf("The loyalty points earned are: %d\n", loyaltyPoints);

    return 0;
}
```

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>

// Function to calculate the total cost of the order
float calculateOrderTotal(float prices[], int numberOfItems) {
    float total = 0.0;
    for (int i = 0; i < numberOfItems; i++) {
        total += prices[i];
    }
```

```
    return total;
}

int main() {
    int numberOfItems;

    // Input 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 (int i = 0; i < numberOfItems; i++) {
        printf("Price of item %d: ", i + 1);
        scanf("%f", &prices[i]);
    }

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

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

    return 0;
}
```

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>

// Function to calculate the refund amount
float calculateRefund(float itemPrice, float refundPercentage) {
    return itemPrice * (refundPercentage / 100);
}

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);

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

    return 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>

// Function to calculate shipping cost based on weight brackets
float calculateShippingCost(float weight) {
    if (weight <= 5.0) {
        return 5.0;
    } else if (weight <= 10.0) {
        return 10.0;
    } else {
        return 15.0;
    }
}

int main() {
    float weight, shippingCost;

    // Input the weight of the package
    printf("Enter the weight of the package (kg): ");
    scanf("%f", &weight);

    shippingCost = calculateShippingCost(weight);

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

    return 0;
}
```

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>

// Function to convert currency
float convertCurrency(float amount, float exchangeRate) {
    return amount * exchangeRate;
}

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

    // Input the amount and exchange rate
    printf("Enter the amount in the original currency: ");
    scanf("%f", &amount);
    printf("Enter the exchange rate: ");
    scanf("%f", &exchangeRate);

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

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

    return 0;
}
```

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>

// Function to find the lower price
float findLowerPrice(float priceA, float priceB) {
    return (priceA < priceB) ? priceA : priceB;
}

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

    // Input the prices from two vendors
    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;
}
```

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>

// Function to check eligibility for senior citizen discount
bool isEligibleForSeniorDiscount(int age) {
    return age >= 60;
}

int main() {
    int age;
    bool eligible;

    // Input the age of the customer
    printf("Enter the age of the customer: ");
    scanf("%d", &age);

    // Check eligibility for senior citizen discount
    eligible = isEligibleForSeniorDiscount(age);

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

    return 0;
}
```