Swap Two Numbers

1. Write a program to swap two numbers using a function. Observe and explain why the original numbers remain unchanged due to call by value.

Without return type

```
#include <stdio.h>


// Function without return type void
swapnum(int a, int b) {
    printf("Swapped value of a is %d:\n",b);
    printf("Swapped value of b is %d:\n",a);
}


int main() {
    int num1, num2;


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


    // Call the function
    swapnum(num1, num2);



    return 0;
}
```

with return type

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


// Function with return type int
swapnum(int a, int b) {
    return a=b;
    return b=a;
}


int main() {
    int num1, num2,swap;


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


    // Call the function and store the result
    swap=swapnum(num1, num2); printf("Swapped
    value of a is %d\n",num2); printf("Swapped value
    of b is %d\n",num1);


    return 0;
}
```

Find Maximum of Two Numbers

2. Implement a function that takes two integers as arguments and returns the larger of the two. Demonstrate how the original values are not altered.

Without return type

```
#include <stdio.h>


// Function without return type void
findLarger(int a, int b) {
    int larger = (a > b) ? a : b;
    printf("The larger number is: %d\n", larger);
}


int main() {
    int num1, num2;


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


    // Call the function
    findLarger(num1, num2);


    return 0;
}
```

With return type

```
#include <stdio.h>


 // Function with return type int
 findLarger(int a, int b) { return
             (a > b) ? a : b;
}


int main() {
    int num1, num2, larger;


    printf("Enter two integers: "); scanf("%d %d",
    &num1, &num2);
    // Call the function and store the result larger

    = findLarger(num1, num2);


    // Original values are not altered   printf("The larger

    number is: %d\n", larger);


    return 0;

}
```

Factorial Calculation

3. 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 fact(int n) { int

    factorial = 1;

    for (int i = 1; i <= n; i++) { factorial

        *= i;

    }

    printf("Factorial of %d is: %d\n", n, factorial);

}


int main() {   int

    number;


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

    if (number < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        fact(number);
    }

    return 0;
}
```

With return type

```
#include <stdio.h>

int fact(int n) {
    int factorial = 1;
    for (int i = 1; i <= n; i++) { factorial
        *= i;
    }
    return factorial;
}

int main() {   int
    number;

    printf("Enter a number: ");
    scanf("%d", &number);
    if (number < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        printf("Factorial of %d is: %d\n", number, fact(number));
    }

    return 0;
}
```

Sum of Two Numbers

4. 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.

Without return type

```
#include <stdio.h>

// Function without return type void
calculateSum(int a, int b) {
    printf("The sum is: %d\n", a + b);
}

int main() {
    int num1, num2;

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

    // Call the function
    calculateSum(num1, num2);

    return 0;
}
```

With return type

```
#include <stdio.h>


// Function with return type  int
calculateSum(int a, int b) {
    return a + b;
}


int main() {
    int num1, num2, sum;


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


    // Call the function and store the result sum =
    calculateSum(num1, num2);


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


    return 0;
}
```

Check Even or Odd

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

Without return type

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


// Function without return type void
oddeven(int num) {
    if(num%2==0){
        printf("The number %d is even\n",num);
    }
    else{
        printf("The number %d is odd\n",num);
    }
}

int main() {   int
    number;

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

    // Call the function
    oddeven(number);

    return 0;
}
```

With return type

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

// Function with return type int
isEven(int num) {
    return (num % 2 == 0) ? 1 : 0; // Return 1 if even, 0 if odd
}

int main() {   int
    number;

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

    // Call the function and check the return value if
    (isEven(number)) {
        printf("%d is even.\n", number);
    }
    else
     {
        printf("%d is odd.\n", number);
    }

    return 0;
}
```

Calculate Simple Interest

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

Without return type

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

// Function to calculate and print simple interest
void cal_interest(float principal, float rate, float time) { float
    interest = (principal * rate * time) / 100; printf("The simple
    interest is: %.2f\n", interest);
}

int main() {
    float principal = 1000; float
    rate = 5;
    float time = 3;

    cal_interest(principal, rate, time);

    return 0;
}
```

With return type

```
#include <stdio.h>

// Function to calculate simple interest
float cal_interest(float principal, float rate, float time) { return
    (principal * rate * time) / 100;
}

int main() {
    float principal = 1000; float
    rate = 5;
    float time = 3;

    float interest = cal_interest(principal, rate, time); printf("The
    simple interest is: %.2f\n", interest);

    return 0;
}
```

Reverse a Number

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

Without return type

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


// Function to reverse an integer (without return type) void
reverseNumber(int n) {
    int reversed = 0, original = n; while
    (n != 0) {
        reversed = reversed * 10 + n % 10; n /= 10;
    }
    printf("Reversed number: %d\n", reversed);
    // Demonstrate that the original number remains unchanged
    printf("Original number: %d\n", original);
}


int main() {   int
    number;


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


    // Call the function
    reverseNumber(number);


    return 0;
}
```

With return type

```
#include <stdio.h>


// Function to reverse an integer (with return type) int
reverseNumber(int n) {
    int reversed = 0; while
    (n != 0) {
        reversed = reversed * 10 + n % 10; n /= 10;
    }
    return reversed;
}


int main() {   int
    number;


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


    // Call the function and display the result
    printf("Reversed number: %d\n", reverseNumber(number));
    // Demonstrate that the original number remains unchanged printf("Original
    number: %d\n", number);


    return 0;
}
```

GCD of Two Numbers

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

Without return type

– – – – – – – – – – –

```
#include <stdio.h>

// Function to calculate GCD (without return type) void
calculateGCD(int a, int b) {
    int originalA = a, originalB = b;
    while (b != 0) {
        int temp = b; b
        = a % b;
        a = temp;
    }
    printf("GCD of %d and %d is: %d\n", originalA, originalB, a);
}

int main() {
    int num1, num2;

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

    // Call the function
    calculateGCD(num1, num2);

    return 0;
}
```

With return type

```
#include <stdio.h>

// Function to calculate GCD (with return type) int
calculateGCD(int a, int b) {
    while (b != 0) {
        int temp = b; b
        = a % b;
        a = temp;
    }
    return a;
}

int main() {
    int num1, num2;

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

    // Call the function and display the result
    printf("GCD of %d and %d is: %d\n", num1, num2, calculateGCD(num1, num2));

    return 0;
}
```

Sum of Digits

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

Without return type

```
#include <stdio.h>


void digitsum(int n) { int
    sum = 0;
    while (n != 0) {
      sum = sum + n % 10; n /=
        10;
    }
    printf("Sum is %d\n",sum);
}


int main() {   int
    number;


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



    digitsum(number);


    return 0;
}
```

With return type

```
#include <stdio.h>


int digitsum(int n) { int
    sum = 0; while (n
    != 0) {
      sum = sum + n % 10; n /=
        10;
    }
    return sum;
}


int main() {   int
    number;


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


    // Call the function and display the result
    printf("Sum of %d is %d\n",number, digitsum(number));


    return 0;
}
```

Prime Number Check

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

Without return type

```
#include <stdio.h>


// Function to check and print if a number is prime void
is_prime(int num) {
    if (num <= 1) {
        printf("%d is not a prime number.\n", num); return;
    }


    for (int i = 2; i * i <= num; i++) { if
        (num % i == 0) {
            printf("%d is not a prime number.\n", num); return;
        }
    }


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


int main() {
    int num = 7;


    is_prime(num); // Directly prints the result


    return 0;
}
```

With return type

```
#include <stdio.h>


// Function to check if a number is prime int
is_prime(int num) {
    if (num <= 1) {
        return 0; // Not prime
    }


    for (int i = 2; i * i <= num; i++) { if
        (num % i == 0) {
            return 0; // Not prime
        }
    }


    return 1; // Prime
}


int main() {
    int num = 7;


    if (is_prime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }
    return 0;
}
```

Fibonacci Sequence Check

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

Without return type

```
#include <stdio.h>

// Function to check if a number is in the Fibonacci sequence (without return type) void
checkFibonacci(int n) {
    int a = 0, b = 1, temp;

    if (n == a || n == b) {
        printf("%d belongs to the Fibonacci sequence.\n", n); return;
    }

    while (b < n) {
        temp = b;  b
        = a + b;  a =
        temp;
    }

    if (b == n) {
        printf("%d belongs to the Fibonacci sequence.\n", n);
    } else {
        printf("%d does not belong to the Fibonacci sequence.\n", n);
    }
}

int main() {   int
    number;

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

    // Call the function
    checkFibonacci(number);

    return 0;
}
```

With return type

```
#include <stdio.h>

// Function to check if a number is in the Fibonacci sequence (with return type) int
isFibonacci(int n) {
    int a = 0, b = 1, temp;

    if (n == a || n == b) {
        return 1; // Return true if the number is 0 or 1
    }
    while (b < n) {
        temp = b;  b
        = a + b;  a =
        temp;
    }

    return (b == n); // Return true if the number matches a Fibonacci number
}

int main() {   int
    number;

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

    // Call the function and display the result if
    (isFibonacci(number)) {
        printf("%d belongs to the Fibonacci sequence.\n", number);
    } else {
        printf("%d does not belong to the Fibonacci sequence.\n", number);
    }

    return 0;
}
```

Quadratic Equation Solver

12. 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.

Without return type

```
#include <stdio.h>


// Function to calculate and print the roots of a quadratic equation void
calculate_roots(int a, int b, int c) {
    int discriminant = b * b - 4 * a * c;  // Calculate discriminant


      if (discriminant > 0) {
          // Two real roots
        printf("The roots are real: %d and %d\n", (-b + discriminant) / (2 * a), (-b - discriminant) / (2 * a));
    } else if (discriminant == 0) {
          // One real root
        printf("The root is: %d\n", -b / (2 * a));
    } else {
        // Complex roots
        printf("The roots are complex.\n");
    }
}


int main() {
    int a = 1, b = -3, c = 2;  // Example coefficients


    calculate_roots(a, b, c);  // Directly prints the roots


    return 0;
}
```

With return type

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

// Function to calculate and print the roots of a quadratic equation void
calculate_roots(int a, int b, int c) {
    int discriminant = b * b - 4 * a * c;  // Calculate discriminant

    if (discriminant > 0) {
        // Two real roots
        printf("The roots are real: %d and %d\n", (-b + discriminant) / (2 * a), (-b - discriminant) / (2 * a));
    } else if (discriminant == 0) {
        // One real root
        printf("The root is: %d\n", -b / (2 * a));
    } else {
        // Complex roots
        printf("The roots are complex.\n");
    }
}

int main() {
    int a = 1, b = -3, c = 2;  // Example coefficients

    calculate_roots(a, b, c);
    return 0;
}
```

Binary to Decimal Conversion

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

Without return type

```
#include <stdio.h>

// Function to convert binary to decimal and print the result void
binary_to_decimal(int binary) {
    int decimal = 0, base = 1, remainder;

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

    printf("Decimal equivalent: %d\n", decimal);
}

int main() {
    int binary = 1011;  // Example binary number
    binary_to_decimal(binary);


    return 0;
}
```

With return type

```
#include <stdio.h>


// Function to convert binary to decimal int
binary_to_decimal(int binary) {
    int decimal = 0, base = 1, remainder;


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


    return decimal;
}


int main() {
    int binary = 1011;  // Example binary number


    printf("Decimal equivalent: %d\n", binary_to_decimal(binary));
    return 0;
}
```

Matrix Trace Calculation

14. 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.

Without return type

```
#include <stdio.h>

// Function to compute and print the trace of a 2x2 matrix void
compute_trace(int a, int b, int c, int d) {
    int trace = a + d;  // Sum of diagonal elements printf("The trace
    of the matrix is: %d\n", trace);
}

int main() {
    int a = 1, b = 2, c = 3, d = 4;

    compute_trace(a, b, c, d);

    return 0;
}
```

With return type

```
#include <stdio.h>


// Function to compute the trace of a 2x2 matrix int
compute_trace(int a, int b, int c, int d) {
    return a + d;  // Sum of diagonal elements
}


int main() {
    int a = 1, b = 2, c = 3, d = 4;


    int trace = compute_trace(a, b, c, d); printf("The trace
    of the matrix is: %d\n", trace);


    return 0;
}
```

Palindrome Number Check

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

Without return type

```
#include <stdio.h>


// Function to check and print whether the number is a palindrome void
is_palindrome(int num) {
    int original = num;
    int reversed = 0, remainder;

    // Reverse the number while
    (num != 0) {
        remainder = num % 10;
        reversed = reversed * 10 + remainder; num /= 10;
    }

    // Check if the original number is equal to the reversed number if (original
    == reversed) {
        printf("%d is a palindrome.\n", original);
    } else {
        printf("%d is not a palindrome.\n", original);
    }
}

int main() {
    int num = 121;
    is_palindrome(num);  // Directly prints the result

    return 0;
}
```

With return type

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

```
// Function to check whether the number is a palindrome int
is_palindrome(int num) {
    int original = num;
    int reversed = 0, remainder;

    // Reverse the number while
    (num != 0) {
        remainder = num % 10;
        reversed = reversed * 10 + remainder; num /= 10;
    }

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

int main() {
    int num = 121;

    if (is_palindrome(num)) {
        printf("%d is a palindrome.\n", num);
    } else {
        printf("%d is not a palindrome.\n", num);
    }
    return 0;
}
```

set 2 questions

----

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 to convert units
float convert_units(float value, char type) { if (type
    == 'C') {
        return value * 0.393701;  // Convert cm to inches
    } else if (type == 'I') {
        return value * 2.54;            // Convert inches to cm
    } else {
        return -1;
    }
}


int main() {
    float value = 5.0; char
    type = 'C';

    float result = convert_units(value, type);

    if (result != -1) {
        printf("Converted value: %.2f\n", result);
    } else {
        printf("Invalid conversion type.\n");
    }

    return 0;
}
```

```
Output
-------
Converted value:1.97
```

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>

// Function to calculate the maximum number of pieces and leftover material int
calculate_cuts(int material_length, int piece_length) {
    if (piece_length == 0) {
        printf("Piece length cannot be zero.\n");
        return -1;  // Return an error value if piece length is zero
    }

    int num_pieces = material_length / piece_length; int leftover
    = material_length % piece_length;

    printf("Maximum number of pieces: %d\n", num_pieces);
    printf("Leftover material: %d\n", leftover);

    return num_pieces;
}

int main() {
    int material_length = 500; int
    piece_length = 210;

    calculate_cuts(material_length, piece_length);

    return 0;
}
```

Output

```
-------
Maximum number of pieces:2 Leftover
material:80
```

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){ if
    (pulley_diameter == 0) {
        printf("Not possible.\n");
        return -1;  // Return an error value if piece length is zero
    }


    float rpm=(belt_speed/(3.14*pulley_diameter))*60;


    printf("The rpm is: %f\n", rpm);


    return rpm;
}


int main() {
    float belt_speed = 20.0; float
    pulley_diameter = 5.0;


    calculate_rpm(belt_speed,pulley_diameter);


    return 0;
}
```

```
Output
--------
The rpm is:76.433
```

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>

// Function to calculate effective production rate
int calculate_production_rate(int speed, int efficiency){ if(speed<0
    || efficiency<0){
        printf("Can't calculate\n"); return -
        1;
    }
    float production_rate=(speed*efficiency)/100; printf("The
    production rate is:%f\n",production_rate); return
    production_rate;

}
int main(){ int
    speed;
    printf("Enter the speed:\n");
    scanf("%d",&speed);
    int efficiency;
    printf("Enter the efficiency:\n");
    scanf("%d",&efficiency);

    calculate_production_rate(speed,efficiency); return 0;



}
```

Output

```
--------
Enter the speed:10 Enter
the efficiency:20
The production rate is:2.000
```

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>

// Function to calculate amount of material wasted
int calculate_wastage(int total_length, int leftover_length){ if(
    total_length==0){
        printf("Calculation not possible");
        return -1;
    }
    float  mat_wasted=total_length-leftover_length;
    printf("The amount of material wasted is:%f\n",mat_wasted); return
    mat_wasted;
}

int main(){
    int total_length;
    printf("Enter the total material length:\n");
    scanf("%d",&total_length);

    int leftover_length ;
    printf("Enter the leftover material length:\n");
    scanf("%d",&leftover_length);

    calculate_wastage(total_length,leftover_length); return 0;



}
```

Output

```
--------
Enter the total material length:100  Enter the
leftover material length:20 The amount of
material wasted is:80.00
```

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 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=5.0; float
    hours=6.0;
    float cost_per_kwh=1.25;

    float energy_cost=calculate_energy_cost(power_rating,hours,cost_per_kwh); printf("Total
    Energy cost is %f\n",energy_cost);
    return 0;


}
```

Output

```
--------
```

```
Total Energy cost is 37.500
```

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>

// Function to calculate heat generated
float calculate_heat(float power_usage, float efficiency){ return
    power_usage*(1-(efficiency/100));
}

int main(){
    float power_usage=10.0; float
    efficiency=6.0;

    float heat_produced=calculate_heat(power_usage,efficiency); printf("Heat
    Generated is %f\n",heat_produced);
    return 0;


}
```

Output

```
--------
Heat Generated is 9.400
```

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>


// Function to calculate wear rate
float calculate_wear_rate(float time, int material_type) { float
    wear_rate;


    // Determine the wear rate factor based on material type if
    (material_type == 1) {
        wear_rate = time * 0.5;
    } else if (material_type == 2) {
        wear_rate = time * 1.0;
    } else {
        printf("Invalid material type.\n"); return -1;
    }


    return wear_rate;
}
```

```
int main() {
    float time = 605.0;
    int material_type = 1;

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

    if (wear_rate != -1) {
        printf("Wear Rate: %.2f%%\n", wear_rate);
    }

    return 0;
}
```

Output

```
-------
Wear Rate:302.50%
```

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 quality
int calculate_reorder_quantity(int consumption_rate, int lead_time) {
    return consumption_rate*lead_time;
}

int main() {
    int consumption_rate = 100.0; int
    lead_time = 30;

    int reorder_quality = calculate_reorder_quantity(consumption_rate,lead_time);

    printf("Reorder Quality is:%d\n",reorder_quality);

    return 0;
}
```

Output

```
-------
Reorder Quality is:3000
```

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 defectiv rate
float calculate_defective_rate(int defective_items, int batch_size) { return
    (defective_items/batch_size)*100;
}

int main() {
    int defective_items= 500;
    int batch_size= 25;

    float result = calculate_defective_rate(defective_items,batch_size);

    printf("Defective Rate is:%f\n",result);

    return 0;
}
```

```
Output
-------
Defective Rate is:2000.00
```

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>

// Function to calculate efficiency
float calculate_efficiency(int output_rate, int downtime) {
    // Total time in minutes is 60 minutes for 1 hour
    return ((60 - downtime) / 60.0) * 100;
}

int main() {
    int output_rate = 96; int
    downtime = 30;

    // Calculate efficiency
    float efficiency = calculate_efficiency(output_rate, downtime);

    printf("Efficiency: %.2f%%\n", efficiency);

    return 0;
}
```

Output

```
-------
50.00%
```

12. Paint Coverage Estimation

Input: Two floating-point numbers: surface area (m²) and paint coverage per liter (m²/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 = 1024.0;
    float coverage = 40;

    float paint_required = calculate_paint(area,coverage);

    printf("Paint Required: %2f liters\n", paint_required);

    return 0;
}
```

```
Output
-------
Paint Required:25.600 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>

// 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 = 250; int
    interval = 10;

    // Calculate hours remaining for maintenance
    int hours_remaining = calculate_maintenance_schedule(current_usage, interval);

    printf("Hours remaining for maintenance: %d hours\n", hours_remaining);

    return 0;
}
```

Output

```
-------
Hours remaining for maintenance:10 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); #include
<stdio.h>

// Function to calculate optimal cycle time in seconds float
calculate_cycle_time(int speed, int operations) {
    return 3600.0 / (speed * operations);
}

int main() {
    int speed = 360; int
    operations = 6;

    float cycle_time = calculate_cycle_time(speed, operations);

    printf("Optimal Cycle Time: %.2f seconds\n", cycle_time);

    return 0;
}
```

Output

```
-------
Optimal Cycle Time:1.67 seconds
```

Set 3 problems

----

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>

// Function to calculate and print the discounted price
void calculateDiscount(float originalPrice, float discountPercentage) {
    float discountedPrice = originalPrice - (originalPrice * discountPercentage / 100);
    printf("Original Price: %.2f\n", originalPrice);
    printf("Discount Percentage: %.2f%%\n", discountPercentage);
    printf("Discounted Price: %.2f\n", discountedPrice);
}

int main() {
    float originalPrice = 150.0;
    float discountPercentage = 35.0;

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

    return 0;
}
```

Output

```
-------
Original Price:150
Discount Percentage:35.0% Discounted
Price:97.50
```

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>
```

```
// Function to calculate the updated inventory count
int updateInventory(int currentCount, int changeQuantity) { return
    currentCount + changeQuantity;
}

int main() {
    int currentCount = 100;
    int changeQuantity = 30;  // Quantity to add (positive) or remove (negative)

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

    printf("Original Inventory Count: %d\n", currentCount); printf("Change
    Quantity: %d\n", changeQuantity); printf("New Inventory Count:
    %d\n", newCount);

    return 0;
}
```

Output

```
-------
Original Inventory Count:100 Change
Quantity:30
New Inventory Count:130
```

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) { return
    itemPrice + (itemPrice * taxRate / 100);
}

int main() {
    float itemPrice = 108.0; float
    taxRate = 12.0;

    float totalPrice = calculateTotalPrice(itemPrice, taxRate);

    printf("Original Price: %.2f\n", itemPrice); printf("Sales Tax
    Rate: %.2f%%\n", taxRate); printf("Total Price After Tax:
    %.2f\n", totalPrice);

    return 0;
}
```

```
Output
--------
Original Price:108.0 Sales
Tax Rate:12.00%
Total Price After Tax:120.96
```

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) {
    return (int)(amountSpent / 10); // 1 point for every $10 spent
}

int main() {
    float amountSpent = 612.50;

    // Call the function to calculate loyalty points
    int loyaltyPoints = calculateLoyaltyPoints(amountSpent);

    printf("Loyalty Points Earned: %d\n", loyaltyPoints);

    return 0;
}
```

Output

```
---------
Loyalty Points:61
```

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() {
    float prices[] = {14.5, 22.0, 2.75, 0.10}; // Array of item prices int
    numberOfItems = 4;

    // Calculate the total cost of the order
    float totalCost = calculateOrderTotal(prices, numberOfItems); printf("Total
    Order Cost: %.2f\n", totalCost);

    return 0;
}
```

Output

```
--------
Total Order Cost:39.35
```

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 = 50.0;

    float refundPercentage = 15.0;


    float refundAmount = calculateRefund(itemPrice, refundPercentage); printf("Refund

    Amount: %.2f\n", refundAmount);


    return 0;

}
```

Output

```
-------

7. 50
```

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 float

calculateShippingCost(float weight) {

    if (weight <= 5) {
        return 5.0;
    } else if (weight <= 10) { return
        10.0;
    } else {
        return 15.0;
    }
}

int main() {
    float weight = 13.2;

    float shippingCost = calculateShippingCost(weight);

    printf("Shipping Cost: $%.2f\n", shippingCost);

    return 0;
}
```

Output

```
--------
Shipping Cost:$15.00
```

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>

// Function to convert currency
```

```
float convertCurrency(float amount, float exchangeRate) { return amount *
    exchangeRate;
}

int main() {
    float amount = 80.0;
    float exchangeRate = 1.4;

    float convertedAmount = convertCurrency(amount, exchangeRate);

    printf("Converted Amount: %.2f\n", convertedAmount);

    return 0;
}
```

Output

```
--------
Converted Amount:112.00
```

9. 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>

// Function to check senior citizen discount eligibility int
isEligibleForSeniorDiscount(int age) {
    return age >= 65; // Return 1 if eligible, 0 otherwise
}

int main() {
    int age = 87;
    // Check eligibility
    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;
}
```

Output

```
-------
The customer is eligible for a senior citizen discount.
```

10. 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 = 300.0; // Price from vendor A float
    priceB = 15.5; // Price from vendor B

    float lowerPrice = findLowerPrice(priceA, priceB);
    printf("Lower Price: %.2f\n", lowerPrice);

    return 0;
}
```

Output

```
Lower Price:15.50
```
