

Document

1. Hello World Program

- Write a program that outputs "Hello, World!" to the console.

```
// Exercise 1

#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

2. Simple Arithmetic Operations

- Create a program that adds, subtracts, multiplies, and divides two numbers input by the user.

```
// Exercise 2

#include <stdio.h>

int main() {
    double num1, num2;

    // Get input from user
    printf("Enter first number: ");
    scanf("%lf", &num1);

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

    // Perform calculations
    printf("\nArithmetic Operations:\n");
    printf("%.2f + %.2f = %.2f\n", num1, num2, num1 + num2);
    printf("%.2f - %.2f = %.2f\n", num1, num2, num1 - num2);
    printf("%.2f * %.2f = %.2f\n", num1, num2, num1 * num2);

    // Check for division by zero
    if (num2 != 0) {
        printf("%.2f / %.2f = %.2f\n", num1, num2, num1 / num2);
    } else {
        printf("Division by zero is not allowed\n");
    }

    return 0;
}
```

3. Even or Odd Checker

- Write a program to check if a given integer is even or odd.

```
// Exercise 3
#include <stdio.h>

int main() {
    int number;

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

    if (number % 2 == 0) {
        printf("%d is even.\n", number);
    } else {
        printf("%d is odd.\n", number);
    }

    return 0;
}
```

4. Largest of Three Numbers

- Develop a program that finds the largest among three numbers entered by the user.

```
// Exercise 4
#include <stdio.h>

int main() {
    double num1, num2, num3, largest;

    // Input three numbers
    printf("Enter three numbers: ");
    scanf("%lf %lf %lf", &num1, &num2, &num3);

    // Method 1: Using if-else statements
    if (num1 >= num2 && num1 >= num3) {
        largest = num1;
    } else if (num2 >= num1 && num2 >= num3) {
        largest = num2;
    } else {
        largest = num3;
    }

    printf("The largest number is: %.2f\n", largest);

    return 0;
}
```

```

#include <stdio.h>

int main() {
    double num1, num2, num3;

    printf("Enter three numbers: ");
    scanf("%lf %lf %lf", &num1, &num2, &num3);

    // Method 2: Using nested ternary operator
    double largest = (num1 >= num2) ?
        (num1 >= num3 ? num1 : num3) :
        (num2 >= num3 ? num2 : num3);

    printf("The largest number is: %.2f\n", largest);

    return 0;
}

```

5. Leap Year Determination

- Create a program to determine whether a given year is a leap year.

```

// Exercise 5
#include <stdio.h>

int main() {
    int year;

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

    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        printf("%d is a leap year.\n", year);
    } else {
        printf("%d is not a leap year.\n", year);
    }

    return 0;
}

```

```

#include <stdio.h>

int main() {
    int year;

    printf("Enter a year: ");
    if (scanf("%d", &year) != 1 || year < 1) {
        printf("Invalid input! Please enter a positive year.\n");
        return 1;
    }
}

```

```

// Check leap year conditions
int isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400

printf("\nYear: %d\n", year);
printf("Result: %s\n", isLeapYear ? "Leap Year" : "Not a Leap Year"
printf("Days in February: %d\n", isLeapYear ? 29 : 28);

return 0;
}

```

6. Sum of Natural Numbers

- Write a program to calculate the sum of the first N natural numbers.

// Exercise 6

```

#include <stdio.h>

int main() {
    int n, sum = 0;

    // Input validation
    printf("Enter a positive integer N: ");
    if (scanf("%d", &n) != 1 || n < 1) {
        printf("Please enter a valid positive integer!\n");
        return 1;
    }

    // Method 1: Using loop
    for (int i = 1; i <= n; i++) {
        sum += i;
    }

    // Display result
    printf("Sum of first %d natural numbers is: %d\n", n, sum);

    // Method 2: Using formula: sum = n(n+1)/2
    int formula_sum = (n * (n + 1)) / 2;
    printf("Using formula: %d\n", formula_sum);

    return 0;
}

```

```

#include <stdio.h>

// Function for recursive calculation
int recursiveSum(int n) {
    if (n == 0)
        return 0;
    return n + recursiveSum(n - 1);
}

```

```

int main() {
    int n;

    printf("Enter a positive integer N: ");
    if (scanf("%d", &n) != 1 || n < 1) {
        printf("Please enter a valid positive integer!\n");
        return 1;
    }

    // Using while loop
    int i = 1, sum1 = 0;
    while (i <= n) {
        sum1 += i;
        i++;
    }

    // Using formula
    int sum2 = (n * (n + 1)) / 2;

    // Using recursion
    int sum3 = recursiveSum(n);

    printf("\nResults using different methods:\n");
    printf("Loop Method: %d\n", sum1);
    printf("Formula Method: %d\n", sum2);
    printf("Recursive Method: %d\n", sum3);

    return 0;
}

```

7. Factorial Calculation

- Develop a program to compute the factorial of a given number.

```

// Exercise 7
#include <stdio.h>

// Function for recursive calculation
long int factorial_recursive(int n) {
    if (n >= 1)
        return n * factorial_recursive(n - 1);
    else
        return 1;
}

int main() {
    int n;
    long int factorial = 1;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

```

```

// Input validation
if (n < 0) {
    printf("Factorial is not defined for negative numbers.\n");
    return 1;
}

// Method 1: Using iterative approach
for (int i = 1; i <= n; i++) {
    factorial *= i;
}

printf("\nFactorial Results:\n");
printf("Iterative Method: %ld\n", factorial);
printf("Recursive Method: %ld\n", factorial_recursive(n));

return 0;
}

```

8. Prime Number Checker

- Write a program to check if a number is a prime number.

```

// Exercise 8

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

int isPrime(int n) {
    // Check for numbers less than 2
    if (n <= 1) {
        return 0;
    }

    // Check for divisibility up to square root
    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int number;

    printf("Enter a positive 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;
}

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

int isPrime(int n) {
    // Handle special cases
    if (n <= 1) return 0;
    if (n <= 3) return 1;
    if (n % 2 == 0 || n % 3 == 0) return 0;

    // Check for divisibility by numbers of form 6k ± 1
    for (int i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int number;

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

    printf("%d is %s prime number.\n",
        number,
        isPrime(number) ? "a" : "not a");

    return 0;
}

```

9. Fibonacci Series

- Create a program to print the Fibonacci series up to N terms.

```

// Exercise 9

#include <stdio.h>

int main() {
    int n, first = 0, second = 1, next;

    printf("Enter number of terms: ");
    scanf("%d", &n);
}

```

```

    if (n < 1) {
        printf("Please enter a positive integer.\n");
        return 1;
    }

    printf("Fibonacci Series: ");

    // Print first term
    printf("%d", first);

    if (n > 1) {
        printf(", %d", second);
    }

    // Generate fibonacci series
    for (int i = 3; i <= n; i++) {
        next = first + second;
        printf(", %d", next);
        first = second;
        second = next;
    }

    printf("\n");
    return 0;
}

```

```

#include <stdio.h>

```

```

int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

```

```

int main() {
    int n;

    printf("Enter number of terms: ");
    scanf("%d", &n);

    if (n < 1) {
        printf("Please enter a positive integer.\n");
        return 1;
    }

    printf("Fibonacci Series: ");
    for (int i = 0; i < n; i++) {
        printf("%d", fibonacci(i));
        if (i < n - 1) {
            printf(", ");
        }
    }
}

```



```
    printf("\n");

    return 0;
}
```

10. Reverse a Number

- Write a program to reverse a given integer number.

// Exercise 10

```
#include <stdio.h>

int main() {
    int num, reversed = 0, remainder;

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

    // Store original number for output
    int original = num;

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

    printf("Original number: %d\n", original);
    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

```
#include <stdio.h>

int reverseNumber(int num, int rev) {
    if (num == 0)
        return rev;

    int remainder = num % 10;
    rev = rev * 10 + remainder;
    return reverseNumber(num / 10, rev);
}

int main() {
    int num;

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

```

    printf("Original number: %d\n", num);
    printf("Reversed number: %d\n", reverseNumber(num, 0));

    return 0;
}

```

11. Palindrome Number

- Develop a program to check if a number is a palindrome.

// Exercise 11

```

#include <stdio.h>

int main() {
    int num, original, reversed = 0, remainder;

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

    original = num;

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

    if (original == reversed) {
        printf("%d is a palindrome.\n", original);
    } else {
        printf("%d is not a palindrome.\n", original);
    }

    return 0;
}

```

```

#include <stdio.h>

// Function to reverse a number
int reverseNumber(int num) {
    int reversed = 0;
    while (num > 0) {
        reversed = reversed * 10 + (num % 10);
        num /= 10;
    }
    return reversed;
}

```

```

// Function to count digits
int countDigits(int num) {
    int count = 0;
    while (num > 0) {
        count++;
        num /= 10;
    }
    return count;
}

int main() {
    int num;

    printf("Enter a positive integer: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Please enter a positive number.\n");
        return 1;
    }

    int digits = countDigits(num);
    int reversed = reverseNumber(num);

    printf("\nNumber Analysis:\n");
    printf("Original number: %d\n", num);
    printf("Reversed number: %d\n", reversed);
    printf("Number of digits: %d\n", digits);
    printf("Result: %d is %s palindrome\n",
           num,
           (num == reversed) ? "a" : "not a");

    return 0;
}

```

12. Armstrong Number Checker

- Write a program to check if a given number is an Armstrong number.

```

// Exercise 12

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

int main() {
    int num, originalNum, remainder, digits = 0;
    double result = 0;

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

    originalNum = num;

```

```

// Count number of digits
while (originalNum != 0) {
    originalNum /= 10;
    digits++;
}

originalNum = num;

// Calculate sum of power of digits
while (originalNum != 0) {
    remainder = originalNum % 10;
    result += pow(remainder, digits);
    originalNum /= 10;
}

if ((int)result == num) {
    printf("%d is an Armstrong number.\n", num);
} else {
    printf("%d is not an Armstrong number.\n", num);
}

return 0;
}

```

```

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

```

```

int countDigits(int num) {
    int count = 0;
    while (num != 0) {
        num /= 10;
        count++;
    }
    return count;
}

```

```

double calculateSum(int num, int digits) {
    double sum = 0;
    while (num != 0) {
        int digit = num % 10;
        sum += pow(digit, digits);
        num /= 10;
    }
    return sum;
}

```

```

int main() {
    int num;

    printf("Enter a positive integer: ");
    scanf("%d", &num);
}

```

```

    if (num < 0) {
        printf("Please enter a positive number.\n");
        return 1;
    }

    int digits = countDigits(num);
    double sum = calculateSum(num, digits);

    printf("\nNumber Analysis:\n");
    printf("Number: %d\n", num);
    printf("Number of digits: %d\n", digits);
    printf("Sum of powers: %.0f\n", sum);
    printf("Result: %d is %s Armstrong number\n",
        num,
        (num == (int)sum) ? "an" : "not an");

    return 0;
}

```

13. Sum of Digits

- Create a program to calculate the sum of digits of a number.

// Exercise 13

```

#include <stdio.h>

int main() {
    int num, sum = 0, digit;

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

    // Store original number for output
    int original = num;

    // Calculate sum of digits
    while (num != 0) {
        digit = num % 10;
        sum += digit;
        num /= 10;
    }

    printf("Sum of digits in %d is: %d\n", original, sum);

    return 0;
}

```

//Enhanced Version with Additional Features

```

#include <stdio.h>

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

// Function to get individual digits
void displayDigits(int num) {
    int digits[20];
    int i = 0;

    // Extract digits in reverse order
    while (num != 0) {
        digits[i++] = num % 10;
        num /= 10;
    }

    printf("Digits: ");
    // Print digits in original order
    for (int j = i-1; j >= 0; j--) {
        printf("%d", digits[j]);
        if (j > 0) printf(" + ");
    }
    printf("\n");
}

int main() {
    int num;

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

    if (num < 0) {
        num = -num;
        printf("Converting to positive number: %d\n", num);
    }

    printf("\nDigit Analysis:\n");
    displayDigits(num);
    printf("Sum = %d\n", sumOfDigits(num));

    return 0;
}

```

14. Swapping Values Without a Temporary Variable

- Write a program to swap two numbers without using a temporary variable.

```

// Exercise 14
#include <stdio.h>

int main() {
    int a, b;

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

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

    // Swapping using addition and subtraction
    a = a + b;    // Store sum in a
    b = a - b;    // Get first number in b
    a = a - b;    // Get second number in a

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

    return 0;
}

//Alternative Method Using Multiplication
#include <stdio.h>

int main() {
    int a, b;

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

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

    // Check for zero to prevent division by zero
    if (b == 0) {
        b = a;
        a = 0;
    } else if (a == 0) {
        a = b;
        b = 0;
    } else {
        a = a * b;    // Store product in a
        b = a / b;    // Get first number in b
        a = a / b;    // Get second number in a
    }
}

```

```

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

    return 0;
}

```

15. ASCII Value Display

- Develop a program to display the ASCII value of a character entered by the user.

// Exercise 15

```

#include <stdio.h>

int main() {
    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    printf("ASCII value of '%c' is: %d\n", ch, ch);

    return 0;
}

```

16. Simple Calculator Using Switch Case

- Create a simple calculator program using switch case statements

// Exercise 16

```

#include <stdio.h>

int main() {
    char operator;
    double num1, num2, result;

    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &operator);

    printf("Enter two numbers: ");
    scanf("%lf %lf", &num1, &num2);

    switch(operator) {
        case '+':
            result = num1 + num2;
            printf("%.2lf + %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '-':
            result = num1 - num2;
            printf("%.2lf - %.2lf = %.2lf\n", num1, num2, result);

```



```

        break;

    case '*':
        result = num1 * num2;
        printf("%.2lf * %.2lf = %.2lf\n", num1, num2, result);
        break;

    case '/':
        if(num2 != 0) {
            result = num1 / num2;
            printf("%.2lf / %.2lf = %.2lf\n", num1, num2, result);
        } else {
            printf("Error: Division by zero!\n");
        }
        break;

    default:
        printf("Error: Invalid operator!\n");
}

return 0;
}

```

//Enhanced Version with Additional Features

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

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

```

void clearInputBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}

```

```

int main() {
    char operator;
    double num1, num2, result;
    int continue_calc = 1;

    while(continue_calc) {
        printf("\nCalculator Menu:\n");
        printf("+ : Addition\n");
        printf("- : Subtraction\n");
        printf("* : Multiplication\n");
        printf("/ : Division\n");
        printf("^ : Power\n");
        printf("r : Square root (of first number)\n");
        printf("q : Quit\n\n");

        printf("Enter operator: ");
        scanf("%c", &operator);
        clearInputBuffer();

        if(operator == 'q') {

```

```

        printf("Calculator closing...\n");
        break;
    }

    if(operator != 'r') {
        printf("Enter first number: ");
        scanf("%lf", &num1);
        printf("Enter second number: ");
        scanf("%lf", &num2);
    } else {
        printf("Enter number: ");
        scanf("%lf", &num1);
    }

    switch(operator) {
        case '+':
            result = num1 + num2;
            printf("\nResult: %.2lf + %.2lf = %.2lf\n", num1, num2,
                break;

        case '-':
            result = num1 - num2;
            printf("\nResult: %.2lf - %.2lf = %.2lf\n", num1, num2,
                break;

        case '*':
            result = num1 * num2;
            printf("\nResult: %.2lf * %.2lf = %.2lf\n", num1, num2,
                break;

        case '/':
            if(num2 != 0) {
                result = num1 / num2;
                printf("\nResult: %.2lf / %.2lf = %.2lf\n", num1, n
            } else {
                printf("\nError: Division by zero!\n");
            }
            break;

        case '^':
            result = pow(num1, num2);
            printf("\nResult: %.2lf ^ %.2lf = %.2lf\n", num1, num2,
                break;

        case 'r':
            if(num1 >= 0) {
                result = sqrt(num1);
                printf("\nResult:  $\sqrt{}$ %.2lf = %.2lf\n", num1, result);
            } else {
                printf("\nError: Cannot calculate square root of ne
            }
            break;
    }

```

```

        default:
            printf("\nError: Invalid operator!\n");
    }

    clearInputBuffer();
}

return 0;
}

```

17. Alphabet Checker

- Write a program to check whether a character is an alphabet or not.

```

// Exercise 17
#include <stdio.h>

int main() {
    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
        printf("'%c' is an alphabet.\n", ch);
    } else {
        printf("'%c' is not an alphabet.\n", ch);
    }

    return 0;
}

//Enhanced Version with Detailed Analysis
#include <stdio.h>
#include <ctype.h>

void analyzeCharacter(char ch) {
    printf("\nCharacter Analysis:\n");
    printf("Character: '%c'\n", ch);

    if(isalpha(ch)) {
        printf("Type: Alphabet\n");
        printf("Case: %s\n", isupper(ch) ? "Uppercase" : "Lowercase");

        if(isupper(ch)) {
            printf("Lowercase equivalent: '%c'\n", tolower(ch));
        } else {
            printf("Uppercase equivalent: '%c'\n", toupper(ch));
        }

        // Check for vowel or consonant
    }
}

```

```

        char lowercaseChar = tolower(ch);
        if(lowercaseChar == 'a' || lowercaseChar == 'e' ||
            lowercaseChar == 'i' || lowercaseChar == 'o' ||
            lowercaseChar == 'u') {
            printf("Category: Vowel\n");
        } else {
            printf("Category: Consonant\n");
        }

    } else {
        printf("Type: Not an alphabet\n");
        if(isdigit(ch)) {
            printf("Category: Digit\n");
        } else if(isspace(ch)) {
            printf("Category: Whitespace\n");
        } else if ispunct(ch)) {
            printf("Category: Punctuation\n");
        } else {
            printf("Category: Special Character\n");
        }
    }

    printf("ASCII Value: %d\n", ch);
}

int main() {
    char ch;

    printf("Enter a character: ");
    ch = getchar();

    analyzeCharacter(ch);

    return 0;
}

```

18. Vowel or Consonant

- Develop a program to check whether a character is a vowel or a consonant.

```

// Exercise 18
#include <stdio.h>
#include <ctype.h>

int main() {
    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    // Convert to lowercase for easier checking
    ch = tolower(ch);

```

```

    if(isalpha(ch)) {
        if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
            printf("'%c' is a vowel.\n", ch);
        else {
            printf("'%c' is a consonant.\n", ch);
        }
    } else {
        printf("'%c' is not an alphabet.\n", ch);
    }

    return 0;
}

```

```

#include <stdio.h>
#include <ctype.h>

```

```

void analyzeCharacter(char ch) {
    printf("\nCharacter Analysis:\n");
    printf("Input Character: '%c'\n", ch);

    if(!isalpha(ch)) {
        printf("Result: Not an alphabet\n");
        return;
    }

    // Convert to lowercase for analysis
    char lowerCh = tolower(ch);

    // Check if vowel
    int isVowel = (lowerCh == 'a' || lowerCh == 'e' ||
                  lowerCh == 'i' || lowerCh == 'o' ||
                  lowerCh == 'u');

    printf("Character Type: %s\n", isVowel ? "Vowel" : "Consonant");
    printf("Case: %s\n", isupper(ch) ? "Uppercase" : "Lowercase");

    if(isVowel) {
        printf("Position in vowel sequence: ");
        switch(lowerCh) {
            case 'a': printf("1st vowel\n"); break;
            case 'e': printf("2nd vowel\n"); break;
            case 'i': printf("3rd vowel\n"); break;
            case 'o': printf("4th vowel\n"); break;
            case 'u': printf("5th vowel\n"); break;
        }
    } else {
        printf("Neighboring Vowels: ");
        // Find closest vowels in alphabet
        char prevVowel = 'a';
        char nextVowel = 'e';
    }
}

```

```

        if(lowerCh > 'u') prevVowel = 'u';
        else if(lowerCh > 'o') prevVowel = 'o';
        else if(lowerCh > 'i') prevVowel = 'i';
        else if(lowerCh > 'e') prevVowel = 'e';
        else if(lowerCh > 'a') prevVowel = 'a';

        if(lowerCh < 'a') nextVowel = 'a';
        else if(lowerCh < 'e') nextVowel = 'e';
        else if(lowerCh < 'i') nextVowel = 'i';
        else if(lowerCh < 'o') nextVowel = 'o';
        else if(lowerCh < 'u') nextVowel = 'u';

        printf("'%c' and '%c'\n", prevVowel, nextVowel);
    }

    printf("ASCII Value: %d\n", ch);
}

int main() {
    char ch;

    printf("Enter a character: ");
    ch = getchar();

    analyzeCharacter(ch);

    return 0;
}

```

19. Number Guessing Game

- Create a simple number guessing game between 1 and 100.

```

// Exercise 19
import random

def number_guessing_game():
    print("Welcome to the Number Guessing Game!")
    print("I have selected a number between 1 and 100. Can you guess it")

    target_number = random.randint(1, 100)
    attempts = 0

    while True:
        try:
            guess = int(input("Enter your guess: "))
            attempts += 1

            if guess < target_number:
                print("Too low! Try again.")
            elif guess > target_number:
                print("Too high! Try again.")

```

```

    else:
        print(f"Congratulations! You guessed the number in {att
        break
except ValueError:
    print("Invalid input. Please enter a number between 1 and 1

```

20. Multiplication Table

- Write a program to display the multiplication table of a given number.

```
#include <stdio.h>

int main() {
    int num, range = 10;

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

    printf("\nMultiplication Table for %d:\n", num);
    printf("-----\n");

    for(int i = 1; i <= range; i++) {
        printf("%d x %2d = %3d\n", num, i, num * i);
    }

    return 0;
}
```

Enhanced Version with Formatting Options

```
#include <stdio.h>

void printTableHeader(int num) {
    printf("\n\n                                \\n");
    printf("    Table of %-14d \\n", num);
    printf("                                \\n");
}

void printTableFooter() {
    printf("                                \\n");
}

void printMultiplicationTable(int num, int start, int end) {
    printTableHeader(num);

    for(int i = start; i <= end; i++) {
        printf("    %2d × %2d = %-11d \\n", num, i, num * i);
    }

    printTableFooter();
}
```

```

}

void printExtendedTable(int num) {
    printf("\nExtended Multiplication Table for %d\n", num);
    printf("
        \n");

    // Print table in grid format
    for(int i = 1; i <= 10; i++) {
        for(int j = 0; j < 5; j++) {
            int multiplier = i + j * 10;
            printf("%2d x %2d = %4d    ", num, multiplier, num * multiplic
        }
        printf("\n");
    }
}

int main() {
    int num, choice;

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

    printf("\nSelect table format:\n");
    printf("1. Basic Table (1-10)\n");
    printf("2. Extended Table (1-50)\n");
    printf("Enter choice (1 or 2): ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printMultiplicationTable(num, 1, 10);
            break;
        case 2:
            printExtendedTable(num);
            break;
        default:
            printf("Invalid choice!\n");
    }

    return 0;
}

```

Sample Outputs

Basic Table Format:

Table of 7

```

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21

```


$7 \times 4 = 28$
 $7 \times 5 = 35$
 $7 \times 6 = 42$
 $7 \times 7 = 49$
 $7 \times 8 = 56$
 $7 \times 9 = 63$
 $7 \times 10 = 70$

Extended Table Format:

Extended Multiplication Table for 7

$7 \times 1 = 7$	$7 \times 11 = 77$	$7 \times 21 = 147$	$7 \times 31 = 217$	7
$7 \times 2 = 14$	$7 \times 12 = 84$	$7 \times 22 = 154$	$7 \times 32 = 224$	7
$7 \times 3 = 21$	$7 \times 13 = 91$	$7 \times 23 = 161$	$7 \times 33 = 231$	7
$7 \times 4 = 28$	$7 \times 14 = 98$	$7 \times 24 = 168$	$7 \times 34 = 238$	7
$7 \times 5 = 35$	$7 \times 15 = 105$	$7 \times 25 = 175$	$7 \times 35 = 245$	7
$7 \times 6 = 42$	$7 \times 16 = 112$	$7 \times 26 = 182$	$7 \times 36 = 252$	7
$7 \times 7 = 49$	$7 \times 17 = 119$	$7 \times 27 = 189$	$7 \times 37 = 259$	7
$7 \times 8 = 56$	$7 \times 18 = 126$	$7 \times 28 = 196$	$7 \times 38 = 266$	7
$7 \times 9 = 63$	$7 \times 19 = 133$	$7 \times 29 = 203$	$7 \times 39 = 273$	7
$7 \times 10 = 70$	$7 \times 20 = 140$	$7 \times 30 = 210$	$7 \times 40 = 280$	7

The enhanced version includes:

- Multiple display formats
- Box drawing characters for better presentation
- Grid layout for extended table
- Proper alignment of numbers
- User choice for table range
- Formatted output with consistent spacing

21. Find GCD of Two Numbers

- Develop a program to find the Greatest Common Divisor (GCD) of two numbers.

```
#include <stdio.h>

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

int main() {
    int num1, num2;

    printf("Enter two positive integers: ");
```

```

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

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

return 0;
}

```

Enhanced Version with Additional Features

```

#include <stdio.h>
#include <stdlib.h>

int gcd(int a, int b) {
    // Use absolute values to handle negative numbers
    a = abs(a);
    b = abs(b);

    // Handle special cases
    if (a == 0) return b;
    if (b == 0) return a;

    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);

    printf("\\nGCD Calculation:\\n");
    printf("Numbers: %d and %d\\n", num1, num2);
    int result = gcd(num1, num2);
    printf("GCD: %d\\n", result);

    // Show that GCD divides both numbers
    printf("\\nVerification:\\n");
    printf("%d ÷ %d = %d (remainder: %d)\\n",
        num1, result, num1/result, num1%result);
    printf("%d ÷ %d = %d (remainder: %d)\\n",
        num2, result, num2/result, num2%result);

    return 0;
}

```

Sample Output

Enter two integers: 48 18

GCD Calculation:

Numbers: 48 and 18

GCD: 6

Verification:

$48 \div 6 = 8$ (remainder: 0)

$18 \div 6 = 3$ (remainder: 0)

The enhanced version includes:

- Handling of negative numbers
- Special case handling for zero
- Verification of the result
- Clear output formatting
- Step-by-step display of the calculation

The program uses the Euclidean algorithm, which states that the GCD of two numbers remains the same if we replace the larger number with its remainder when divided by the smaller number.

22. Find LCM of Two Numbers

- Write a program to find the Least Common Multiple (LCM) of two numbers.

```
#include <stdio.h>

int main() {
    int n1, n2, i, gcd, lcm;

    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);

    // Calculate GCD first
    for (i = 1; i <= n1 && i <= n2; ++i) {
        if (n1 % i == 0 && n2 % i == 0)
            gcd = i;
    }

    // Calculate LCM using the formula: LCM = (n1 * n2) / GCD
    lcm = (n1 * n2) / gcd;

    printf("The LCM of %d and %d is %d.", n1, n2, lcm);
    return 0;
}
```

How It Works

The program uses the relationship between LCM and GCD of two numbers to calculate the result efficiently[2]. The formula used is:

$$\text{LCM}(a,b) = \frac{a \times b}{\text{GCD}(a,b)}$$

Key Steps:

1. Take two numbers as input from the user
2. Calculate their GCD using a loop
3. Apply the LCM formula
4. Display the result

For example, if you input the numbers 72 and 120, the program will output 360 as their LCM.

Alternative Approach

You could also use a while loop method, which works by:

1. Finding the maximum of the two numbers
2. Incrementally checking multiples until finding one divisible by both numbers
3. This approach is less efficient but more straightforward to understand.

The GCD method is recommended as it requires fewer iterations and is more computationally

23. Count Number of Digits

- Create a program to count the number of digits in an integer.

```
// Exercise 23
#include <stdio.h>
#include <math.h>

int main() {
    int number, count = 0;

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

    // Store the number in a temporary variable
    int temp = abs(number);

    // Handle the case when number is 0
    if (number == 0) {
        count = 1;
    }

    // Count digits using division method
    while (temp != 0) {
        count++;
        temp = temp / 10;
    }
}
```

```

        printf("Number of digits: %d\n", count);
        return 0;
    }

// OR:
#include <stdio.h>
#include <math.h>

int main() {
    int number;

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

    // Handle negative numbers
    number = abs(number);

    // Using logarithm method
    int digits = number == 0 ? 1 : (int)log10(number) + 1;

    printf("Number of digits: %d\n", digits);
    return 0;
}

```

24. Power of a Number Using Loop

- Develop a program to calculate the power of a number using a loop.

```

#include <stdio.h>

int main() {
    int base, exponent;
    long long result = 1;

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

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

    // Calculate power using loop
    for(int i = 1; i <= exponent; i++) {
        result = result * base;
    }

    printf("%d^%d = %lld", base, exponent, result);
    return 0;
}

```

How It Works

The program uses a simple multiplication loop to calculate the power. For example, if base = 3 and exponent = 4, the calculation would be:

1. First iteration: $\text{result} = 1 \times 3 = 3$
2. Second iteration: $\text{result} = 3 \times 3 = 9$
3. Third iteration: $\text{result} = 9 \times 3 = 27$
4. Fourth iteration: $\text{result} = 27 \times 3 = 81$

Alternative Method

You can also use a while loop approach:

```
#include <stdio.h>

int main() {
    int base, exponent;
    long double result = 1.0;

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

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

    while(exponent != 0) {
        result *= base;
        --exponent;
    }

    printf("Answer = %.0Lf", result);
    return 0;
}
```

Key Points:

- The program handles positive exponents
- Uses long long or long double to handle larger results
- The loop multiplies the base number by itself exponent times
- Time complexity is $O(n)$ where n is the exponent

For production code, consider using the built-in `pow()` function from the math library for better precision and handling of negative exponents.

25. Binary to Decimal Conversion

- Write a program to convert a binary number to decimal.

```
// Exercise 25
```

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

```
int main() {
```

```

int binary, decimal = 0, base = 1, remainder;

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

// Store original number for output
int temp = binary;

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

printf("%d in binary = %d in decimal", temp, decimal);
return 0;
}

```

26. Decimal to Binary Conversion

- Create a program to convert a decimal number to binary.

```

// Exercise 26

#include <stdio.h>

int main() {
    int decimal, binary[32], i = 0;

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

    // Store original number for output
    int temp = decimal;

    // Convert decimal to binary
    while (decimal > 0) {
        binary[i] = decimal % 2;
        decimal = decimal / 2;
        i++;
    }

    printf("%d in binary is: ", temp);
    // Print binary array in reverse order
    for (int j = i - 1; j >= 0; j--) {
        printf("%d", binary[j]);
    }

    return 0;
}

```

```

#include <stdio.h>

int main() {
    int decimal;

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

    printf("%d in binary is: ", decimal);

    // Handle the case when decimal is 0
    if (decimal == 0) {
        printf("0");
        return 0;
    }

    // Find the highest set bit
    int bits = sizeof(decimal) * 8;
    int leading_zeros = 1;

    for (int i = bits - 1; i >= 0; i--) {
        int bit = (decimal >> i) & 1;
        if (bit == 1) {
            leading_zeros = 0;
        }
        if (!leading_zeros) {
            printf("%d", bit);
        }
    }

    return 0;
}

```

27. Sum of Array Elements

- Write a program to find the sum of all elements in an array.

```

// Exercise 27

#include <stdio.h>

int main() {
    int size;

    printf("Enter the size of array: ");
    scanf("%d", &size);

    int arr[size];

    // Input array elements

```



```

printf("Enter %d elements:\n", size);
for(int i = 0; i < size; i++) {
    scanf("%d", &arr[i]);
}

// Calculate sum using for loop
int sum = 0;
for(int i = 0; i < size; i++) {
    sum += arr[i];
}

printf("Sum of array elements: %d", sum);
return 0;
}

```

28. Largest Element in an Array

- Develop a program to find the largest element in an array.

```

#include <stdio.h>

int main() {
    int size;

    printf("Enter the size of array: ");
    scanf("%d", &size);

    int arr[size];

    // Input array elements
    printf("Enter %d elements:\\n", size);
    for(int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Find largest element
    int max = arr[0]; // Assume first element is largest
    for(int i = 1; i < size; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
    }

    printf("Largest element: %d", max);
    return 0;
}

```

How It Works

The program uses a simple linear search approach:

1. Initialize the first element as maximum
2. Compare each element with current maximum
3. Update maximum if a larger element is found
4. Continue until all elements are checked

Alternative Methods

Using While Loop:

```
#include <stdio.h>

int main() {
    int size;
    printf("Enter the size of array: ");
    scanf("%d", &size);

    int arr[size];
    printf("Enter %d elements:\\n", size);
    for(int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    int max = arr[0];
    int i = 1;
    while(i < size) {
        if(arr[i] > max) {
            max = arr[i];
        }
        i++;
    }

    printf("Largest element: %d", max);
    return 0;
}
```

Using Recursive Function:

```
#include <stdio.h>

int findMax(int arr[], int n) {
    if (n == 1)
        return arr[0];

    int max = findMax(arr, n-1);
    return (max > arr[n-1]) ? max : arr[n-1];
}

int main() {
    int size;
    printf("Enter the size of array: ");
    scanf("%d", &size);
```

```

    int arr[size];
    printf("Enter %d elements:\\n", size);
    for(int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Largest element: %d", findMax(arr, size));
    return 0;
}

```

Key Points:

- Time complexity is $O(n)$ for all methods
- The iterative method is most memory efficient
- The recursive method uses additional stack space
- Program assumes array has at least one element
- Works with both positive and negative integers

For best performance, use the iterative approach with a for loop. It's simple, efficient, and doesn't require extra memory like the recursive solution.

29. Array Sorting in Ascending Order

- Write a program to sort an array in ascending order.

```

// Exercise 29
#include <stdio.h>

int main() {
    int size;

    printf("Enter the size of array: ");
    scanf("%d", &size);

    int arr[size];

    // Input array elements
    printf("Enter %d elements:\\n", size);
    for(int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Bubble sort implementation
    for(int i = 0; i < size - 1; i++) {
        for(int j = 0; j < size - i - 1; j++) {
            if(arr[j] > arr[j + 1]) {
                // Swap elements
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

```

```

printf("Sorted array: ");
for(int i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}

return 0;
}

```

30. Matrix Addition

- Create a program to add two matrices of size 2x2.

// Exercise 30

```

#include <stdio.h>

int main() {
    int mat1[2][2], mat2[2][2], result[2][2];

    // Input first matrix
    printf("Enter elements of first matrix (2x2):\n");
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%d", &mat1[i][j]);
        }
    }

    // Input second matrix
    printf("\nEnter elements of second matrix (2x2):\n");
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%d", &mat2[i][j]);
        }
    }

    // Add matrices
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }

    // Display result
    printf("\nResultant Matrix:\n");
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }
}

```

```
    }

    return 0;
}
```

31. Matrix Multiplication

- Develop a program to multiply two matrices

```
#include <stdio.h>

int main() {
    int r1, c1, r2, c2;

    printf("Enter rows and columns for first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and columns for second matrix: ");
    scanf("%d %d", &r2, &c2);

    // Check if multiplication is possible
    if (c1 != r2) {
        printf("Matrix multiplication not possible!");
        return 1;
    }

    int mat1[r1][c1], mat2[r2][c2], result[r1][c2];

    // Input first matrix
    printf("\nEnter elements of first matrix:\n");
    for(int i = 0; i < r1; i++) {
        for(int j = 0; j < c1; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }

    // Input second matrix
    printf("\nEnter elements of second matrix:\n");
    for(int i = 0; i < r2; i++) {
        for(int j = 0; j < c2; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }

    // Initialize result matrix with zeros
    for(int i = 0; i < r1; i++) {
        for(int j = 0; j < c2; j++) {
            result[i][j] = 0;
        }
    }

    // Perform matrix multiplication
    for(int i = 0; i < r1; i++) {
```

```

        for(int j = 0; j < c2; j++) {
            for(int k = 0; k < c1; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }

    // Display result
    printf("\nResultant Matrix:\n");
    for(int i = 0; i < r1; i++) {
        for(int j = 0; j < c2; j++) {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}

```

How It Works

The matrix multiplication follows these steps:

1. Check if matrices are compatible (columns of first = rows of second)
2. Initialize result matrix with zeros
3. Multiply row elements of first matrix with column elements of second matrix
4. Sum the products for each position in result matrix

For example, multiplying two 2×2 matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} p & q \\ r & s \end{pmatrix} = \begin{pmatrix} ap + br & aq + bs \\ cp + dr & cq + ds \end{pmatrix}$$

Key Points

- Matrix multiplication is only possible when columns of first matrix equal rows of second matrix
- The resulting matrix has dimensions: (rows of first) \times (columns of second)
- Time complexity is $O(n^3)$ for $n \times n$ matrices
- Matrix multiplication is not commutative ($AB \neq BA$)
- For large matrices, consider using optimized algorithms like Strassen's algorithm
- The program assumes sufficient memory for matrix storage

32. Transpose of a Matrix

- Write a program to find the transpose of a matrix.

```
// Exercise 32
```

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

```

int main() {
    int rows, cols;

    printf("Enter number of rows: ");
    scanf("%d", &rows);
    printf("Enter number of columns: ");
    scanf("%d", &cols);

    int matrix[rows][cols], transpose[cols][rows];

    // Input matrix elements
    printf("\nEnter matrix elements:\n");
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }

    // Calculate transpose
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            transpose[j][i] = matrix[i][j];
        }
    }

    // Display original matrix
    printf("\nOriginal Matrix:\n");
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }

    // Display transpose
    printf("\nTranspose Matrix:\n");
    for(int i = 0; i < cols; i++) {
        for(int j = 0; j < rows; j++) {
            printf("%d\t", transpose[i][j]);
        }
        printf("\n");
    }

    return 0;
}

```

33. String Length Calculation Without strlen()

- Create a program to compute the length of a string without using the `strlen()` function.

```
// Exercise 33
```

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

```
int main() {
    char str[100];
    int length = 0;

    printf("Enter a string: ");
    gets(str); // Note: gets() is unsafe, used here for simplicity

    // Calculate length using while loop
    while(str[length] != '\0') {
        length++;
    }

    printf("Length of string: %d", length);
    return 0;
}
```

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

```
int main() {
    char str[100];
    int length;

    printf("Enter a string: ");
    gets(str);

    for(length = 0; str[length] != '\0'; length++);

    printf("Length of string: %d", length);
    return 0;
}
```

34. String Palindrome Checker

- Develop a program to check if a string is a palindrome.

```
// Exercise 34
```

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

```
int main() {
    char str[100];
    int left = 0, isPalindrome = 1;

    printf("Enter a string: ");
    scanf("%s", str);

    // Find string length
    int right = 0;
```



```

while(str[right] != '\0') {
    right++;
}
right--; // Move to last character

// Check palindrome
while(left < right) {
    if(str[left] != str[right]) {
        isPalindrome = 0;
        break;
    }
    left++;
    right--;
}

if(isPalindrome)
    printf("%s is a palindrome", str);
else
    printf("%s is not a palindrome", str);

return 0;
}

```

35. String Concatenation Without strcat()

- Write a program to concatenate two strings without using the `strcat()` function.

// Exercise 35

```

#include <stdio.h>

int main() {
    char str1[100], str2[50];
    int i = 0, j = 0;

    printf("Enter first string: ");
    scanf("%s", str1);

    printf("Enter second string: ");
    scanf("%s", str2);

    // Find the end of first string
    while(str1[i] != '\0') {
        i++;
    }

    // Copy second string to end of first string
    while(str2[j] != '\0') {
        str1[i] = str2[j];
        i++;
        j++;
    }
}

```

```

    // Add null terminator
    str1[i] = '\0';

    printf("Concatenated string: %s", str1);
    return 0;
}

```

36. Count Vowels and Consonants in a String

- Create a program to count the number of vowels and consonants in a string.

```

// Exercise 36
#include <stdio.h>
#include <ctype.h>

int main() {
    char str[100];
    int vowels = 0, consonants = 0;

    printf("Enter a string: ");
    gets(str); // Note: gets() is unsafe, used for simplicity

    for(int i = 0; str[i] != '\0'; i++) {
        // Convert to lowercase for easier checking
        char ch = tolower(str[i]);

        // Check if character is an alphabet
        if(isalpha(ch)) {
            if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
                vowels++;
            else
                consonants++;
        }
    }

    printf("Vowels: %d\n", vowels);
    printf("Consonants: %d", consonants);

    return 0;
}

```

37. Swapping Numbers Using Pointers

- Write a program to swap two numbers using pointer variables.

```

// Exercise 37

#include <stdio.h>

void swap(int *a, int *b) {

```

```

    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int num1, num2;

    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);

    printf("\nBefore swapping:");
    printf("\nFirst number = %d", num1);
    printf("\nSecond number = %d", num2);

    swap(&num1, &num2);

    printf("\n\nAfter swapping:");
    printf("\nFirst number = %d", num1);
    printf("\nSecond number = %d", num2);

    return 0;
}

```

38. Structure Usage: Student Information

- Develop a program to define a structure for a student (name, age, grade) and display the information.

```

// Exercise 38
#include <stdio.h>
#include <string.h>

struct Student {
    char name[50];
    int age;
    float grade;
};

int main() {
    struct Student student1;

    // Input student information
    printf("Enter student name: ");
    gets(student1.name);

    printf("Enter student age: ");
    scanf("%d", &student1.age);

    printf("Enter student grade: ");

```

```

scanf("%f", &student1.grade);

// Display student information
printf("\nStudent Information:\n");
printf("Name: %s\n", student1.name);
printf("Age: %d\n", student1.age);
printf("Grade: %.2f\n", student1.grade);

return 0;
}

```

39. File Handling: Write User Input to a File

- Write a program to create a text file and write user input to the file.

```

// Exercise 39
#include <stdio.h>

int main() {
    FILE *file;
    char text[100];

    // Open file in write mode
    file = fopen("output.txt", "w");

    if(file == NULL) {
        printf("Error opening file!");
        return 1;
    }

    printf("Enter text to write to file (press Enter to finish):\n");
    gets(text);

    // Write text to file
    fprintf(file, "%s", text);

    // Close file
    fclose(file);

    printf("\nText has been written to file successfully.");
    return 0;
}

#include <stdio.h>

int main() {
    FILE *file;
    char ch;

    file = fopen("output.txt", "w");

```

```

    if(file == NULL) {
        printf("Error opening file!");
        return 1;
    }

    printf("Enter text (press Ctrl+Z to finish):\n");

    while((ch = getchar()) != EOF) {
        fputc(ch, file);
    }

    fclose(file);
    printf("\nText written successfully.");
    return 0;
}

```

40. Command Line Arguments Demonstration

- Create a program to demonstrate the use of command line arguments by printing them out.

```

// Exercise 40
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Number of arguments: %d\n", argc);

    printf("\nArguments passed:\n");
    for(int i = 0; i < argc; i++) {
        printf("argv[%d]: %s\n", i, argv[i]);
    }

    return 0;
}

```

1. Positive, Negative, or Zero

- Write a program that reads an integer from the user and determines whether it is positive, negative, or zero.

```

// Exercise 1

#include <stdio.h>

int main() {
    int number;

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

```

```

    if (number > 0) {
        printf("%d is positive\n", number);
    }
    else if (number < 0) {
        printf("%d is negative\n", number);
    }
    else {
        printf("The number is zero\n");
    }

    return 0;
}

```

2. Grade Calculator

- Create a program that takes a score (0-100) as input and outputs the corresponding grade:

```

A (90-100)
B (80-89)
C (70-79)
D (60-69)
F (0-59)

```

```

// Exercise 2
#include <stdio.h>

int main() {
    int score;

    printf("Enter the score (0-100): ");
    scanf("%d", &score);

    if (score < 0 || score > 100) {
        printf("Invalid score! Please enter a score between 0 and 100\n");
        return 1;
    }

    if (score >= 90) {
        printf("Grade: A\n");
    }
    else if (score >= 80) {
        printf("Grade: B\n");
    }
    else if (score >= 70) {
        printf("Grade: C\n");
    }
    else if (score >= 60) {
        printf("Grade: D\n");
    }
    else {

```

```
        printf("Grade: F\n");
    }

    return 0;
}
```

3. Eligibility for Voting

- Write a program that asks the user for their age and determines if they are eligible to vote (18 years or older).

```
// Exercise 3
#include <stdio.h>

int main() {
    int age;

    printf("Enter your age: ");
    scanf("%d", &age);

    if (age < 0) {
        printf("Invalid age! Age cannot be negative\n");
        return 1;
    }

    if (age >= 18) {
        printf("You are eligible to vote!\n");
    }
    else {
        printf("You are not eligible to vote yet\n");
        printf("You will be eligible to vote in %d years\n", 18 - age);
    }

    return 0;
}
```

4. Triangle Type Checker

- Develop a program that takes three angles as input and determines if they form an acute, obtuse, or right triangle.

```
// Exercise 4

#include <stdio.h>

int main() {
    int angle1, angle2, angle3, sum;

    printf("Enter three angles of the triangle: ");
```

```

scanf("%d %d %d", &angle1, &angle2, &angle3);

sum = angle1 + angle2 + angle3;

if (sum != 180 || angle1 <= 0 || angle2 <= 0 || angle3 <= 0) {
    printf("Invalid triangle! The angles must be positive and sum t
    return 1;
}

if (angle1 == 90 || angle2 == 90 || angle3 == 90) {
    printf("This is a right triangle\n");
}
else if (angle1 > 90 || angle2 > 90 || angle3 > 90) {
    printf("This is an obtuse triangle\n");
}
else {
    printf("This is an acute triangle\n");
}

return 0;
}

```

5. Leap Year Determination with Input Validation

- Create a program that prompts the user to enter a year and checks if it's a leap year. Include input validation to ensure the year is a positive integer.

```

// Exercise 5
#include <stdio.h>

int main() {
    int year;

    printf("Enter a year: ");
    if (scanf("%d", &year) != 1 || year <= 0) {
        printf("Invalid input! Please enter a positive year\n");
        return 1;
    }

    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        printf("%d is a leap year\n", year);
    }
    else {
        printf("%d is not a leap year\n", year);
    }

    return 0;
}

```

6. Password Authentication

- Write a program that asks the user to enter a password and checks if it matches a predefined password. Allow up to three attempts.

// Exercise 6

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

int main() {
    const char correct_password[] = "pass123";
    char entered_password[50];
    int attempts = 0;
    const int MAX_ATTEMPTS = 3;

    while (attempts < MAX_ATTEMPTS) {
        printf("Enter password (Attempt %d/%d): ", attempts + 1, MAX_ATTEMPTS);
        scanf("%s", entered_password);

        if (strcmp(entered_password, correct_password) == 0) {
            printf("Access granted!\n");
            return 0;
        }

        printf("Incorrect password. ");
        if (attempts < MAX_ATTEMPTS - 1) {
            printf("Please try again.\n");
        }

        attempts++;
    }

    printf("Maximum attempts reached. Access denied!\n");
    return 1;
}
```

7. Quadratic Equation Solver

- Develop a program that takes coefficients (a) , (b) , and (c) of a quadratic equation $(ax^2 + bx + c = 0)$ and calculates the roots. Handle real and imaginary roots appropriately.

// Exercise 7

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

int main() {
    double a, b, c, discriminant, realPart, imagPart;
```

```

printf("Enter coefficients (a b c): ");
scanf("%lf %lf %lf", &a, &b, &c);

if (a == 0) {
    printf("This is not a quadratic equation\n");
    return 1;
}

discriminant = b*b - 4*a*c;

if (discriminant > 0) {
    double root1 = (-b + sqrt(discriminant)) / (2*a);
    double root2 = (-b - sqrt(discriminant)) / (2*a);
    printf("Two distinct real roots:\n");
    printf("x1 = %.2f\n", root1);
    printf("x2 = %.2f\n", root2);
}
else if (discriminant == 0) {
    double root = -b / (2*a);
    printf("One repeated real root:\n");
    printf("x = %.2f\n", root);
}
else {
    realPart = -b / (2*a);
    imagPart = sqrt(-discriminant) / (2*a);
    printf("Complex roots:\n");
    printf("x1 = %.2f + %.2fi\n", realPart, imagPart);
    printf("x2 = %.2f - %.2fi\n", realPart, imagPart);
}

return 0;
}

```

8. Even Numbers Sum and Average

- Write a program that reads numbers from the user until they enter zero. Calculate and display the sum and average of all even numbers entered.

```

// Exercise 8
#include <stdio.h>

int main() {
    int number, sum = 0, count = 0;
    double average;

    printf("Enter numbers (0 to stop):\n");

    while (1) {
        scanf("%d", &number);

        if (number == 0) {

```

```

        break;
    }

    if (number % 2 == 0) {
        sum += number;
        count++;
    }
}

if (count == 0) {
    printf("No even numbers were entered\n");
}
else {
    average = (double)sum / count;
    printf("\nResults for even numbers:\n");
    printf("Sum: %d\n", sum);
    printf("Count: %d\n", count);
    printf("Average: %.2f\n", average);
}

return 0;
}

```

9. Simple ATM Transaction

- Create a program that simulates an ATM. It should prompt the user for a PIN code and allow them to perform withdrawals if the PIN is correct. Include balance checking and insufficient funds warnings.

```

// Exercise 9
#include <stdio.h>

int main() {
    const int CORRECT_PIN = 1234;
    double balance = 1000.00;
    int pin, attempts = 0;
    int choice;
    double amount;

    printf("Welcome to the ATM\n");

    while (attempts < 3) {
        printf("Enter your PIN: ");
        scanf("%d", &pin);

        if (pin == CORRECT_PIN) {
            break;
        }

        attempts++;
        if (attempts < 3) {

```

```

        printf("Incorrect PIN. Please try again (%d attempts remain
    }
}

if (attempts == 3) {
    printf("Too many incorrect attempts. Card blocked.\n");
    return 1;
}

do {
    printf("\n=== ATM Menu ===\n");
    printf("1. Check Balance\n");
    printf("2. Withdraw Money\n");
    printf("3. Exit\n");
    printf("Enter choice (1-3): ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Current balance: $%.2f\n", balance);
            break;

        case 2:
            printf("Enter amount to withdraw: $");
            scanf("%lf", &amount);

            if (amount <= 0) {
                printf("Invalid amount\n");
            }
            else if (amount > balance) {
                printf("Insufficient funds\n");
            }
            else {
                balance -= amount;
                printf("Please take your cash\n");
                printf("Remaining balance: $%.2f\n", balance);
            }
            break;

        case 3:
            printf("Thank you for using our ATM\n");
            return 0;

        default:
            printf("Invalid choice\n");
    }
} while (1);

return 0;
}

```

10. Body Mass Index (BMI) Calculator

- Write a program that calculates the BMI based on user input for weight in kilograms and height in meters. Output their BMI category (Underweight, Normal weight, Overweight, Obesity).

// Exercise 10

```
#include <stdio.h>

int main() {
    float weight, height, bmi;

    printf("Enter weight (kg): ");
    if (scanf("%f", &weight) != 1 || weight <= 0) {
        printf("Invalid weight! Please enter a positive number\n");
        return 1;
    }

    printf("Enter height (m): ");
    if (scanf("%f", &height) != 1 || height <= 0) {
        printf("Invalid height! Please enter a positive number\n");
        return 1;
    }

    bmi = weight / (height * height);

    printf("\nBMI Results:\n");
    printf("Your BMI: %.1f\n", bmi);
    printf("Category: ");

    if (bmi < 18.5) {
        printf("Underweight\n");
    }
    else if (bmi < 25.0) {
        printf("Normal weight\n");
    }
    else if (bmi < 30.0) {
        printf("Overweight\n");
    }
    else {
        printf("Obesity\n");
    }

    return 0;
}
```

11. Temperature Conversion Menu

- Develop a program that provides a menu to convert temperatures between Celsius, Fahrenheit, and Kelvin based on user choice.

```

// Exercise 11

#include <stdio.h>

int main() {
    int choice;
    float temperature, converted;

    printf("Temperature Conversion Menu:\n");
    printf("1. Celsius to Fahrenheit\n");
    printf("2. Celsius to Kelvin\n");
    printf("3. Fahrenheit to Celsius\n");
    printf("4. Fahrenheit to Kelvin\n");
    printf("5. Kelvin to Celsius\n");
    printf("6. Kelvin to Fahrenheit\n");
    printf("Enter choice (1-6): ");

    if (scanf("%d", &choice) != 1 || choice < 1 || choice > 6) {
        printf("Invalid choice! Please select 1-6\n");
        return 1;
    }

    printf("Enter temperature: ");
    if (scanf("%f", &temperature) != 1) {
        printf("Invalid temperature!\n");
        return 1;
    }

    switch (choice) {
        case 1: // C to F
            converted = (temperature * 9/5) + 32;
            printf("%.2f°C = %.2f°F\n", temperature, converted);
            break;

        case 2: // C to K
            converted = temperature + 273.15;
            printf("%.2f°C = %.2f K\n", temperature, converted);
            break;

        case 3: // F to C
            converted = (temperature - 32) * 5/9;
            printf("%.2f°F = %.2f°C\n", temperature, converted);
            break;

        case 4: // F to K
            converted = (temperature - 32) * 5/9 + 273.15;
            printf("%.2f°F = %.2f K\n", temperature, converted);
            break;

        case 5: // K to C
            if (temperature < 0) {
                printf("Error: Kelvin cannot be negative!\n");
                return 1;
            }
    }
}

```

```

    }
    converted = temperature - 273.15;
    printf("%.2f K = %.2f°C\n", temperature, converted);
    break;

case 6: // K to F
    if (temperature < 0) {
        printf("Error: Kelvin cannot be negative!\n");
        return 1;
    }
    converted = (temperature - 273.15) * 9/5 + 32;
    printf("%.2f K = %.2f°F\n", temperature, converted);
    break;
}

return 0;
}

```

12. Number Divisibility Checker

- Write a program that takes an integer input and checks if it is divisible by both 5 and 11.

```

// Exercise 12
#include <stdio.h>

int main() {
    int number;

    printf("Enter a number: ");
    if (scanf("%d", &number) != 1) {
        printf("Invalid input! Please enter an integer\n");
        return 1;
    }

    printf("\nDivisibility Check for %d:\n", number);

    if (number % 5 == 0 && number % 11 == 0) {
        printf("%d is divisible by both 5 and 11\n", number);
        printf("5 goes in %d times\n", number / 5);
        printf("11 goes in %d times\n", number / 11);
    }
    else {
        if (number % 5 == 0) {
            printf("%d is divisible by 5 but not by 11\n", number);
        }
        else if (number % 11 == 0) {
            printf("%d is divisible by 11 but not by 5\n", number);
        }
        else {
            printf("%d is not divisible by either 5 or 11\n", number);
        }
    }
}

```

```

        printf("Remainder when divided by 5: %d\n", number % 5);
        printf("Remainder when divided by 11: %d\n", number % 11);
    }
}

return 0;
}

```

13. Triangle Validity Checker

- Create a program that reads the lengths of three sides and determines if they can form a valid triangle.

// Exercise 13

```

#include <stdio.h>

int main() {
    float side1, side2, side3;

    printf("Enter the lengths of three sides:\n");
    printf("Side 1: ");
    if (scanf("%f", &side1) != 1 || side1 <= 0) {
        printf("Invalid input! Length must be positive\n");
        return 1;
    }

    printf("Side 2: ");
    if (scanf("%f", &side2) != 1 || side2 <= 0) {
        printf("Invalid input! Length must be positive\n");
        return 1;
    }

    printf("Side 3: ");
    if (scanf("%f", &side3) != 1 || side3 <= 0) {
        printf("Invalid input! Length must be positive\n");
        return 1;
    }

    printf("\nTriangle Analysis:\n");

    // Triangle Inequality Theorem check
    if (side1 + side2 > side3 &&
        side2 + side3 > side1 &&
        side1 + side3 > side2) {

        printf("This is a valid triangle!\n");

        // Determine triangle type by sides
        if (side1 == side2 && side2 == side3) {
            printf("Type: Equilateral triangle\n");

```



```

    }
    else if (side1 == side2 || side2 == side3 || side1 == side3) {
        printf("Type: Isosceles triangle\n");
    }
    else {
        printf("Type: Scalene triangle\n");
    }

    // Calculate perimeter
    float perimeter = side1 + side2 + side3;
    printf("Perimeter: %.2f\n", perimeter);
}
else {
    printf("These sides cannot form a triangle!\n");
    printf("Remember: The sum of any two sides must be greater than
}

return 0;
}

```

14. Day of the Week

- Write a program that takes a number (1-7) as input and displays the corresponding day of the week.

// Exercise 14

```

#include <stdio.h>

int main() {
    int day;

    printf("Enter a number (1-7): ");
    if (scanf("%d", &day) != 1) {
        printf("Invalid input! Please enter a number\n");
        return 1;
    }

    printf("\nDay Analysis:\n");

    switch (day) {
        case 1:
            printf("Day %d is Monday\n", day);
            printf("It's the start of the work week\n");
            break;

        case 2:
            printf("Day %d is Tuesday\n", day);
            break;

        case 3:

```

```

        printf("Day %d is Wednesday\n", day);
        printf("It's the middle of the work week\n");
        break;

    case 4:
        printf("Day %d is Thursday\n", day);
        break;

    case 5:
        printf("Day %d is Friday\n", day);
        printf("It's the end of the work week\n");
        break;

    case 6:
        printf("Day %d is Saturday\n", day);
        printf("It's the weekend!\n");
        break;

    case 7:
        printf("Day %d is Sunday\n", day);
        printf("It's the weekend!\n");
        break;

    default:
        printf("Invalid day number!\n");
        printf("Please enter a number between 1 and 7\n");
        return 1;
}

// Additional information about work/weekend days
if (day >= 1 && day <= 5) {
    printf("This is a weekday\n");
}
else if (day == 6 || day == 7) {
    printf("This is a weekend day\n");
}

return 0;
}

```

15. Electricity Bill Calculator

- Develop a program that calculates the electricity bill based on the total units consumed. Use the following tari :

For the first 50 units: \$0.50 per unit
 For the next 100 units: \$0.75 per unit
 For the next 100 units: \$1.20 per unit
 For units above 250: \$1.50 per unit
 Add a 20% surcharge to the total bill.

```
// Exercise 15
#include <stdio.h>

int main() {
    float units, bill, surcharge, total_bill;

    printf("Enter units consumed: ");
    if (scanf("%f", &units) != 1 || units < 0) {
        printf("Invalid input! Units must be positive\n");
        return 1;
    }

    if (units <= 50) {
        bill = units * 0.50;
    }
    else if (units <= 150) {
        bill = (50 * 0.50) + ((units - 50) * 0.75);
    }
    else if (units <= 250) {
        bill = (50 * 0.50) + (100 * 0.75) + ((units - 150) * 1.20);
    }
    else {
        bill = (50 * 0.50) + (100 * 0.75) + (100 * 1.20) + ((units - 250) * 1.50);
    }

    surcharge = bill * 0.20;
    total_bill = bill + surcharge;

    printf("\nElectricity Bill Breakdown:\n");
    printf("Units Consumed: %.2f\n", units);
    printf("Base Bill: $%.2f\n", bill);
    printf("Surcharge (20%%): $%.2f\n", surcharge);
    printf("Total Bill: $%.2f\n", total_bill);

    return 0;
}
```

16. Number Comparison

- Write a program that reads two integers and determines whether the first is greater than, less than, or equal to the second.

```
// Exercise 16
#include <stdio.h>

int main() {
    int num1, num2;

    printf("Enter first number: ");
    if (scanf("%d", &num1) != 1) {
        printf("Invalid input! Please enter an integer\n");
    }
}
```

```

        return 1;
    }

    printf("Enter second number: ");
    if (scanf("%d", &num2) != 1) {
        printf("Invalid input! Please enter an integer\n");
        return 1;
    }

    printf("\nComparison Results:\n");

    if (num1 > num2) {
        printf("%d is greater than %d\n", num1, num2);
        printf("Difference: %d\n", num1 - num2);
    }
    else if (num1 < num2) {
        printf("%d is less than %d\n", num1, num2);
        printf("Difference: %d\n", num2 - num1);
    }
    else {
        printf("%d is equal to %d\n", num1, num2);
    }

    // Additional analysis
    printf("\nNumber Properties:\n");

    if ((num1 % 2 == 0) && (num2 % 2 == 0)) {
        printf("Both numbers are even\n");
    }
    else if ((num1 % 2 != 0) && (num2 % 2 != 0)) {
        printf("Both numbers are odd\n");
    }
    else {
        printf("One number is even and one is odd\n");
    }

    if (num1 * num2 > 0) {
        printf("Both numbers have the same sign\n");
    }
    else if (num1 * num2 < 0) {
        printf("Numbers have opposite signs\n");
    }
    else {
        printf("At least one number is zero\n");
    }

    return 0;
}

```

17. Character Case Converter

- Create a program that takes a single character as input and converts it to the opposite case (upper to lower, lower to upper).

```
// Exercise 17
#include <stdio.h>
#include <ctype.h>

int main() {
    char ch;

    printf("Enter a character: ");
    ch = getchar();

    if (!isalpha(ch)) {
        printf("Error: Not a letter! Please enter an alphabetic character\n");
        return 1;
    }

    printf("\nCharacter Analysis:\n");
    printf("Original character: '%c'\n", ch);

    if (isupper(ch)) {
        printf("Converted to lowercase: '%c'\n", tolower(ch));
        printf("ASCII value: %d\n", ch);
        printf("Position in alphabet: %d\n", ch - 'A' + 1);
    }
    else {
        printf("Converted to uppercase: '%c'\n", toupper(ch));
        printf("ASCII value: %d\n", ch);
        printf("Position in alphabet: %d\n", ch - 'a' + 1);
    }

    // Additional character information
    printf("\nCharacter Properties:\n");
    if (isalpha(ch)) printf("    Alphabetic\n");
    if (isupper(ch)) printf("    Uppercase\n");
    if (islower(ch)) printf("    Lowercase\n");
    if (isalnum(ch)) printf("    Alphanumeric\n");
    if (isprint(ch)) printf("    Printable\n");

    return 0;
}
```

18. Age Category Determination

- Write a program that asks for a person's age and outputs their life stage:

Child (0-12)
Teenager (13-19)
Adult (20-64)
Senior (65 and above)

```

// Exercise 18
#include <stdio.h>

int main() {
    int age;

    printf("Enter age: ");
    if (scanf("%d", &age) != 1) {
        printf("Invalid input! Please enter a number\n");
        return 1;
    }

    if (age < 0) {
        printf("Invalid age! Age cannot be negative\n");
        return 1;
    }

    printf("\nAge Analysis:\n");
    printf("Age: %d years\n", age);
    printf("Life Stage: ");

    if (age <= 12) {
        printf("Child\n");
        if (age <= 5) {
            printf("Subcategory: Early Childhood\n");
        }
        else {
            printf("Subcategory: School Age\n");
        }
    }
    else if (age <= 19) {
        printf("Teenager\n");
        if (age <= 16) {
            printf("Subcategory: Early Teen\n");
        }
        else {
            printf("Subcategory: Late Teen\n");
        }
    }
    else if (age <= 64) {
        printf("Adult\n");
        if (age <= 35) {
            printf("Subcategory: Young Adult\n");
        }
        else if (age <= 50) {
            printf("Subcategory: Middle Adult\n");
        }
        else {
            printf("Subcategory: Mature Adult\n");
        }
    }
    else {
        printf("Senior\n");
    }
}

```

```

        if (age <= 75) {
            printf("Subcategory: Young Senior\n");
        }
        else if (age <= 85) {
            printf("Subcategory: Middle Senior\n");
        }
        else {
            printf("Subcategory: Advanced Senior\n");
        }
    }

    // Additional age-related information
    printf("\nAdditional Information:\n");

    if (age >= 18) {
        printf("    Legal adult in most countries\n");
    }
    if (age >= 21) {
        printf("    Legal adult in all countries\n");
    }
    if (age >= 65) {
        printf("    Retirement age in many countries\n");
    }

    return 0;
}

```

19. Simple Interest Calculator

- Develop a program that calculates simple interest based on user input for principal amount, rate of interest, and time period.

```

// Exercise 19
#include <stdio.h>

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

    printf("Enter principal amount ($): ");
    if (scanf("%f", &principal) != 1 || principal < 0) {
        printf("Invalid input! Principal cannot be negative\n");
        return 1;
    }

    printf("Enter rate of interest (%): ");
    if (scanf("%f", &rate) != 1 || rate < 0) {
        printf("Invalid input! Rate cannot be negative\n");
        return 1;
    }

    printf("Enter time period (years): ");

```

```

if (scanf("%f", &time) != 1 || time < 0) {
    printf("Invalid input! Time cannot be negative\n");
    return 1;
}

interest = (principal * rate * time) / 100;

printf("\nSimple Interest Calculation:\n");
printf("Principal Amount: $%.2f\n", principal);
printf("Rate of Interest: %.2f%%\n", rate);
printf("Time Period: %.1f years\n", time);
printf("Simple Interest: $%.2f\n", interest);
printf("Total Amount: $%.2f\n", principal + interest);

// Additional analysis
printf("\nLoan Analysis:\n");
printf("Monthly Payment: $%.2f\n", (principal + interest) / (time * 12));
printf("Interest to Principal Ratio: %.2f%%\n", (interest / principal) * 100);

return 0;
}

```

20. Discount Calculator

- Write a program that calculates the discount and final price for a product based on the purchase amount:

If amount > \$1000, apply a 10% discount.

// Exercise 20

```

#include <stdio.h>

int main() {
    float amount, discount = 0, final_price;

    printf("Enter purchase amount ($): ");
    if (scanf("%f", &amount) != 1 || amount < 0) {
        printf("Invalid input! Amount cannot be negative\n");
        return 1;
    }

    if (amount > 1000) {
        discount = amount * 0.10;
    }

    final_price = amount - discount;

    printf("\nDiscount Calculation:\n");
    printf("Purchase Amount: $%.2f\n", amount);
    printf("Discount (10%%): $%.2f\n", discount);
}

```



```

printf("Final Price: $%.2f\n", final_price);

// Additional analysis
if (discount > 0) {
    printf("\nSavings Analysis:\n");
    printf("You saved: %.1f%% of your purchase\n", (discount/amount
    printf("Effective price per dollar: $%.2f\n", final_price/amoun
}

return 0;
}

```

21. Character Type Identifier

- Create a program that identifies whether the entered character is a vowel, consonant, digit, or special character.

```

// Exercise 21
#include <stdio.h>
#include <ctype.h>

int main() {
    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    if (isdigit(ch)) {
        printf("'%.c' is a digit\n", ch);
    }
    else if (isalpha(ch)) {
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == '
        ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == '
        printf("'%.c' is a vowel\n", ch);
    } else {
        printf("'%.c' is a consonant\n", ch);
    }
}
else {
    printf("'%.c' is a special character\n", ch);
}

return 0;
}

```

22. Quadrant Finder

- Write a program that reads the coordinates (x, y) of a point and determines which quadrant it lies in.

```
// Exercise 22
def read_coordinates():
    x = float(input("Enter the x-coordinate: "))
    y = float(input("Enter the y-coordinate: "))
    return x, y

def find_quadrant(x, y):
    if x > 0 and y > 0:
        return "Quadrant I"
    elif x < 0 and y > 0:
        return "Quadrant II"
    elif x < 0 and y < 0:
        return "Quadrant III"
    elif x > 0 and y < 0:
        return "Quadrant IV"
    elif x == 0 and y != 0:
        return "On the Y-axis"
    elif y == 0 and x != 0:
        return "On the X-axis"
    else:
        return "At the Origin"

# Main program
x, y = read_coordinates()
result = find_quadrant(x, y)
print(f"The point ({x}, {y}) is in {result}")
```

23. Maximum and Minimum of Five Numbers

- Develop a program that reads five integers and finds the maximum and minimum values among them.

```
// Exercise 23
#include <stdio.h>

int main() {
    int numbers[5];
    int max, min;

    // Input five numbers
    printf("Enter five integers:\n");
    for(int i = 0; i < 5; i++) {
        printf("Number %d: ", i + 1);
        scanf("%d", &numbers[i]);
    }

    // Initialize max and min with first number
    max = min = numbers[0];

    // Find max and min
    for(int i = 1; i < 5; i++) {
```

```

        if(numbers[i] > max) {
            max = numbers[i];
        }
        if(numbers[i] < min) {
            min = numbers[i];
        }
    }

    printf("\nMaximum value: %d\n", max);
    printf("Minimum value: %d\n", min);

    return 0;
}

```

24. Currency Denomination Breakdown

- Write a program that takes an amount in dollars and provides the least number of notes (100, 50, 20, 10, 5, 1) that sum up to the amount.

```

// Exercise 24
#include <stdio.h>

int main() {
    int amount, notes;
    int denominations[] = {100, 50, 20, 10, 5, 1};
    int size = 6;

    printf("Enter amount: ");
    scanf("%d", &amount);
    printf("\nBreakdown of %d:\n", amount);

    for (int i = 0; i < size; i++) {
        notes = amount / denominations[i];
        if (notes > 0) {
            printf("%d note(s) of $%d\n", notes, denominations[i]);
            amount = amount % denominations[i];
        }
    }

    return 0;
}

```

25. Student Grade Evaluator

- Create a program that reads marks of five subjects and calculates the percentage and grade according to the following criteria:

Percentage	90%: Grade A
Percentage	80%: Grade B

Percentage 70%: Grade C
Percentage 60%: Grade D
Percentage 40%: Grade E
Below 40%: Grade F

```
// Exercise 25
#include <stdio.h>

int main() {
    float marks[5], total = 0, percentage;
    char grade;

    // Input marks for five subjects
    printf("Enter marks for 5 subjects (out of 100 each):\n");
    for(int i = 0; i < 5; i++) {
        printf("Subject %d: ", i + 1);
        scanf("%f", &marks[i]);

        // Validate marks
        if(marks[i] < 0 || marks[i] > 100) {
            printf("Invalid marks! Please enter marks between 0 and 100\n");
            return 1;
        }
        total += marks[i];
    }

    // Calculate percentage
    percentage = total / 5;

    // Determine grade
    if(percentage >= 90) {
        grade = 'A';
    } else if(percentage >= 80) {
        grade = 'B';
    } else if(percentage >= 70) {
        grade = 'C';
    } else if(percentage >= 60) {
        grade = 'D';
    } else if(percentage >= 40) {
        grade = 'E';
    } else {
        grade = 'F';
    }

    // Display results
    printf("\nResults:\n");
    printf("Total Marks: %.2f/500\n", total);
    printf("Percentage: %.2f%%\n", percentage);
    printf("Grade: %c\n", grade);

    return 0;
}
```

26. Calculator with Basic Operations

- Write a program that acts as a simple calculator. It should ask the user for two numbers and an operator (+, -, *, /) and perform the corresponding operation.

```
// Exercise 26
#include <stdio.h>

int main() {
    double num1, num2, result;
    char operator;

    // Input first number
    printf("Enter first number: ");
    scanf("%lf", &num1);

    // Input operator
    printf("Enter operator (+, -, *, /): ");
    scanf(" %c", &operator);

    // Input second number
    printf("Enter second number: ");
    scanf("%lf", &num2);

    // Perform calculation based on operator
    switch(operator) {
        case '+':
            result = num1 + num2;
            printf("%.2lf + %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '-':
            result = num1 - num2;
            printf("%.2lf - %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '*':
            result = num1 * num2;
            printf("%.2lf * %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '/':
            if(num2 != 0) {
                result = num1 / num2;
                printf("%.2lf / %.2lf = %.2lf\n", num1, num2, result);
            } else {
                printf("Error: Division by zero!\n");
            }
            break;

        default:
            printf("Error: Invalid operator!\n");
    }
}
```

```
    }

    return 0;
}
```

27. Year to Century Converter

- Develop a program that converts a given year into its corresponding century.

```
// Exercise 27
#include <stdio.h>

int main() {
    int year, century;

    // Input year
    printf("Enter a year: ");
    scanf("%d", &year);

    // Validate input
    if (year <= 0) {
        printf("Error: Please enter a valid year (greater than 0)\n");
        return 1;
    }

    // Calculate century
    century = (year + 99) / 100;

    // Display result with appropriate suffix
    printf("Year %d is in the ", year);

    // Add appropriate suffix (st, nd, rd, th)
    if (century % 10 == 1 && century % 100 != 11) {
        printf("%dst", century);
    } else if (century % 10 == 2 && century % 100 != 12) {
        printf("%dnd", century);
    } else if (century % 10 == 3 && century % 100 != 13) {
        printf("%drd", century);
    } else {
        printf("%dth", century);
    }
    printf(" century\n");

    return 0;
}
```

28. Triangle Area Calculator

- Write a program that calculates the area of a triangle given its three sides using Heron's

formula. Validate if the sides can form a triangle before calculation.

```
// Exercise 28
#include <stdio.h>
#include <math.h>

int main() {
    double a, b, c, s, area;

    // Input the sides
    printf("Enter three sides of the triangle:\n");
    scanf("%lf %lf %lf", &a, &b, &c);

    // Validate the triangle
    if (a + b <= c || a + c <= b || b + c <= a) {
        printf("These sides cannot form a triangle!\n");
        return 1;
    }

    // Calculate semi-perimeter
    s = (a + b + c) / 2;

    // Calculate area using Heron's formula
    area = sqrt(s * (s - a) * (s - b) * (s - c));

    printf("Area of the triangle: %.2lf square units\n", area);

    return 0;
}
```

29. Root Classification

- Create a program that reads a number and determines if it is a perfect square, perfect cube, both, or neither.

```
// Exercise 29
#include <stdio.h>
#include <math.h>

int main() {
    int num;
    int isSquare = 0, isCube = 0;

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

    // Check for perfect square
    float sqRoot = sqrt((double)num);
    int sqInt = sqRoot;
    if(sqInt == sqRoot) {
        isSquare = 1;
    }
}
```

```

    }

    // Check for perfect cube
    int cubeRoot = round(pow(num, 1.0/3.0));
    if(cubeRoot * cubeRoot * cubeRoot == num) {
        isCube = 1;
    }

    // Output classification
    printf("%d is ", num);
    if(isSquare && isCube) {
        printf("both a perfect square and a perfect cube");
    } else if(isSquare) {
        printf("a perfect square");
    } else if(isCube) {
        printf("a perfect cube");
    } else {
        printf("neither a perfect square nor a perfect cube");
    }
    printf("\n");

    return 0;
}

```

30. Time Converter

- Write a program that converts time from 24-hour format to 12-hour format with AM/PM notation.

```

// Exercise 30
#include <stdio.h>

int main() {
    int hours, minutes;

    // Input time
    printf("Enter time in 24-hour format (HH:MM): ");
    scanf("%d:%d", &hours, &minutes);

    // Validate input
    if (hours < 0 || hours > 23 || minutes < 0 || minutes > 59) {
        printf("Invalid time format!\n");
        return 1;
    }

    // Convert and display time
    if (hours == 0) {
        printf("12:%02d AM\n", minutes);
    }
    else if (hours < 12) {
        printf("%d:%02d AM\n", hours, minutes);
    }
}

```



```

    }
    else if (hours == 12) {
        printf("12:%02d PM\n", minutes);
    }
    else {
        printf("%d:%02d PM\n", hours - 12, minutes);
    }

    return 0;
}

```

31. Profit or Loss Calculator

- Develop a program that calculates profit or loss based on the cost price and selling price entered by the user.

// Exercise 31

```

#include <stdio.h>

int main() {
    float costPrice, sellingPrice;
    float difference, percentage;

    // Input prices
    printf("Enter Cost Price: ");
    scanf("%f", &costPrice);

    printf("Enter Selling Price: ");
    scanf("%f", &sellingPrice);

    // Validate input
    if (costPrice <= 0 || sellingPrice < 0) {
        printf("Error: Prices must be positive!\n");
        return 1;
    }

    // Calculate difference and percentage
    difference = sellingPrice - costPrice;
    percentage = (difference / costPrice) * 100;

    // Display results
    printf("\nResults:\n");
    if (difference > 0) {
        printf("Profit: %.2f\n", difference);
        printf("Profit Percentage: %.2f%%\n", percentage);
    }
    else if (difference < 0) {
        printf("Loss: %.2f\n", -difference);
        printf("Loss Percentage: %.2f%%\n", -percentage);
    }
}

```

```

    else {
        printf("No Profit No Loss\n");
    }

    return 0;
}

```

32. Valid Date Checker

- Write a program that takes day, month, and year as input and checks if it forms a valid date.

```

// Exercise 32
#include <stdio.h>

int main() {
    int day, month, year;
    int isValid = 1;
    int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // Input date
    printf("Enter date (DD MM YYYY): ");
    scanf("%d %d %d", &day, &month, &year);

    // Basic validation
    if(year < 1 || month < 1 || month > 12 || day < 1) {
        isValid = 0;
    }

    // Check for leap year
    if(month == 2) {
        if((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            daysInMonth[1] = 29;
        }
    }

    // Validate day for the given month
    if(day > daysInMonth[month-1]) {
        isValid = 0;
    }

    // Display result
    if(isValid) {
        printf("%02d/%02d/%04d is a valid date\n", day, month, year);

        // Additional information about the date
        if(month == 2 && daysInMonth[1] == 29) {
            printf("This is a leap year\n");
        }
    } else {
        printf("Invalid date!\n");
    }
}

```

```
    }  
  
    return 0;  
}
```

33. Multiple Conditions Checker

- Create a program that checks if an input number is positive and divisible by both 3 and 7.

```
// Exercise 33  
#include <stdio.h>  
  
int main() {  
    int number;  
  
    // Input number  
    printf("Enter a number: ");  
    scanf("%d", &number);  
  
    // Check all conditions  
    printf("\nResults for number %d:\n", number);  
  
    if(number > 0) {  
        if(number % 3 == 0 && number % 7 == 0) {  
            printf("    Number is positive\n");  
            printf("    Number is divisible by both 3 and 7\n");  
            printf("This number meets all conditions!\n");  
        }  
        else if(number % 3 == 0) {  
            printf("    Number is positive\n");  
            printf("    Number is divisible by 3\n");  
            printf("    Number is not divisible by 7\n");  
        }  
        else if(number % 7 == 0) {  
            printf("    Number is positive\n");  
            printf("    Number is not divisible by 3\n");  
            printf("    Number is divisible by 7\n");  
        }  
        else {  
            printf("    Number is positive\n");  
            printf("    Number is not divisible by 3\n");  
            printf("    Number is not divisible by 7\n");  
        }  
    }  
    else {  
        printf("    Number is not positive\n");  
    }  
  
    return 0;  
}
```

34. BMI Status with Input Validation

- Write a program similar to Exercise 10 but include input validation to ensure weight and height are positive numbers.

```
// Exercise 34
#include <stdio.h>

int main() {
    float weight, height, bmi;

    // Get and validate weight
    do {
        printf("Enter weight (in kg): ");
        if(scanf("%f", &weight) != 1 || weight <= 0) {
            printf("Error: Weight must be a positive number!\n");
            while(getchar() != '\n'); // Clear input buffer
            continue;
        }
        break;
    } while(1);

    // Get and validate height
    do {
        printf("Enter height (in meters): ");
        if(scanf("%f", &height) != 1 || height <= 0) {
            printf("Error: Height must be a positive number!\n");
            while(getchar() != '\n'); // Clear input buffer
            continue;
        }
        break;
    } while(1);

    // Calculate BMI
    bmi = weight / (height * height);

    // Display BMI and category
    printf("\nBMI Results:\n");
    printf("Your BMI: %.1f\n", bmi);
    printf("Category: ");

    if(bmi < 18.5) {
        printf("Underweight\n");
    } else if(bmi < 25.0) {
        printf("Normal weight\n");
    } else if(bmi < 30.0) {
        printf("Overweight\n");
    } else {
        printf("Obese\n");
    }
}
```

```

    // Additional health information
    printf("\nBMI Categories:\n");
    printf("< 18.5      : Underweight\n");
    printf("18.5 - 24.9: Normal weight\n");
    printf("25.0 - 29.9: Overweight\n");
    printf("≥ 30.0      : Obese\n");

    return 0;
}

```

35. Character Encryption

- Develop a program that takes a character input and outputs the next character in the ASCII sequence.

```

// Exercise 35
#include <stdio.h>

int main() {
    char input, encrypted;

    // Get character input
    printf("Enter a character: ");
    scanf(" %c", &input);

    // Perform encryption
    if((input >= 'a' && input <= 'z') || (input >= 'A' && input <= 'Z'))
        if(input == 'z')
            encrypted = 'a';
        else if(input == 'Z')
            encrypted = 'A';
        else
            encrypted = input + 1;

    printf("\nOriginal character: %c\n", input);
    printf("ASCII value: %d\n", input);
    printf("Encrypted character: %c\n", encrypted);
    printf("New ASCII value: %d\n", encrypted);
} else {
    printf("\nPlease enter a valid alphabetic character!\n");
}

return 0;
}

```

36. Speeding Ticket Calculator

- Write a program that calculates the fine for speeding based on the speed limit and the

driver's speed:

Up to 10 mph over limit: \$50

11-20 mph over limit: \$100

Over 20 mph: \$200

// Exercise 36

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

```
int main() {
    int speedLimit, actualSpeed, difference;

    // Input speeds
    printf("Enter speed limit (mph): ");
    scanf("%d", &speedLimit);

    printf("Enter actual speed (mph): ");
    scanf("%d", &actualSpeed);

    // Validate input
    if(speedLimit <= 0 || actualSpeed < 0) {
        printf("Error: Please enter valid speeds!\n");
        return 1;
    }

    // Calculate speed difference
    difference = actualSpeed - speedLimit;

    // Display results
    printf("\nSpeed Analysis:\n");
    printf("Speed Limit: %d mph\n", speedLimit);
    printf("Your Speed: %d mph\n", actualSpeed);

    if(difference <= 0) {
        printf("Status: No violation - Speed is within limit\n");
    } else {
        printf("Violation: %d mph over limit\n", difference);
        printf("Fine: $");

        // Calculate fine
        if(difference <= 10) {
            printf("50 (Minor violation)\n");
        } else if(difference <= 20) {
            printf("100 (Moderate violation)\n");
        } else {
            printf("200 (Major violation)\n");
        }
    }

    return 0;
}
```

37. Insurance Premium Calculator

- Create a program that calculates insurance premiums based on age and health conditions:

Age < 25 and good health: Low premium
Age ≥ 25 or poor health: High premium

```
// Exercise 37
#include <stdio.h>

int main() {
    int age;
    char health;
    float basePremium = 500.0;
    float finalPremium;

    // Get age
    printf("Enter age: ");
    scanf("%d", &age);

    // Validate age
    if(age <= 0 || age > 120) {
        printf("Error: Please enter a valid age!\n");
        return 1;
    }

    // Get health status
    printf("Enter health status (G for Good, P for Poor): ");
    scanf(" %c", &health);

    // Convert to uppercase for consistency
    health = toupper(health);

    // Validate health input
    if(health != 'G' && health != 'P') {
        printf("Error: Invalid health status! Use G or P.\n");
        return 1;
    }

    // Calculate premium
    printf("\nInsurance Premium Calculation:\n");
    printf("Age: %d years\n", age);
    printf("Health Status: %s\n", (health == 'G') ? "Good" : "Poor");

    if(age < 25 && health == 'G') {
        finalPremium = basePremium;
        printf("Category: Low Premium\n");
    } else {
        finalPremium = basePremium * 1.5;
        printf("Category: High Premium\n");
    }
}
```

```

// Add age-based adjustments
if(age >= 50) {
    finalPremium *= 1.2;
    printf("Age Factor: 20%% increase applied\n");
}

// Display final premium
printf("\nBase Premium: $%.2f\n", basePremium);
printf("Final Premium: $%.2f\n", finalPremium);

return 0;
}

```

38. Gross Salary Calculation

- Write a program that computes the gross salary of an employee based on basic salary:

HRA = 20% of basic if basic \leq \$10000, else 15%
 DA = 80% of basic if basic \leq \$10000, else 90%

```

// Exercise 38
#include <stdio.h>

int main() {
    float basicSalary, hra, da, grossSalary;

    // Input basic salary
    printf("Enter basic salary: $");
    scanf("%f", &basicSalary);

    // Validate input
    if(basicSalary <= 0) {
        printf("Error: Basic salary must be positive!\n");
        return 1;
    }

    // Calculate HRA and DA based on conditions
    if(basicSalary <= 10000) {
        hra = basicSalary * 0.20; // 20% of basic
        da = basicSalary * 0.80; // 80% of basic
    } else {
        hra = basicSalary * 0.15; // 15% of basic
        da = basicSalary * 0.90; // 90% of basic
    }

    // Calculate gross salary
    grossSalary = basicSalary + hra + da;

    // Display detailed breakdown
    printf("\nSalary Breakdown:\n");
}

```



```

printf("=====\n");
printf("Basic Salary: $%.2f\n", basicSalary);
printf("HRA      : $%.2f ", hra);
printf("(%.0f%%)\n", (basicSalary <= 10000) ? 20.0 : 15.0);
printf("DA      : $%.2f ", da);
printf("(%.0f%%)\n", (basicSalary <= 10000) ? 80.0 : 90.0);
printf("=====\n");
printf("Gross Salary: $%.2f\n", grossSalary);

// Display additional information
printf("\nComponent Percentages:\n");
printf("Basic: %.1f%%\n", (basicSalary/grossSalary) * 100);
printf("HRA  : %.1f%%\n", (hra/grossSalary) * 100);
printf("DA   : %.1f%%\n", (da/grossSalary) * 100);

return 0;
}

```

39. Number Classification

- Develop a program that reads an integer and classifies it as

Single-digit
 Double-digit
 Triple-digit
 More than three digits

```

// Exercise 39
#include <stdio.h>
#include <stdlib.h>

int main() {
    int number, temp;
    int digitCount = 0;

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

    // Handle negative numbers
    temp = abs(number);

    // Count digits
    if(temp == 0) {
        digitCount = 1;
    } else {
        while(temp > 0) {
            digitCount++;
            temp /= 10;
        }
    }
}

```

```

// Display results
printf("\nNumber Analysis:\n");
printf("Number: %d\n", number);
printf("Number of digits: %d\n", digitCount);
printf("Classification: ");

switch(digitCount) {
    case 1:
        printf("Single-digit number\n");
        printf("Range: -9 to 9\n");
        break;

    case 2:
        printf("Double-digit number\n");
        printf("Range: -99 to 99\n");
        break;

    case 3:
        printf("Triple-digit number\n");
        printf("Range: -999 to 999\n");
        break;

    default:
        printf("More than three digits\n");
        printf("Range: Beyond ±999\n");
}

// Additional information
printf("\nDetailed Information:\n");
if(number < 0) {
    printf("Sign: Negative\n");
} else if(number > 0) {
    printf("Sign: Positive\n");
} else {
    printf("Sign: Zero\n");
}

return 0;
}

```

40. Voting System Simulation

- Write a program that simulates a simple voting system. It should allow users to vote for candidates (A, B, C) by entering their choice. Count and display the total votes for each candidate after all votes are cast.

```

// Exercise 40
#include <stdio.h>

int main() {

```

```

int votes_A = 0, votes_B = 0, votes_C = 0;
char vote;
int total_votes = 0;

printf("Simple Voting System\n");
printf("-----\n");
printf("Candidates:\n");
printf("A - Candidate A\n");
printf("B - Candidate B\n");
printf("C - Candidate C\n");
printf("X - End Voting\n\n");

while(1) {
    printf("Enter your vote (A/B/C or X to end): ");
    scanf(" %c", &vote);

    // Convert to uppercase for consistency
    vote = toupper(vote);

    if(vote == 'X') {
        break;
    }

    switch(vote) {
        case 'A':
            votes_A++;
            total_votes++;
            break;
        case 'B':
            votes_B++;
            total_votes++;
            break;
        case 'C':
            votes_C++;
            total_votes++;
            break;
        default:
            printf("Invalid vote! Please vote A, B, or C\n");
            continue;
    }

    printf("Vote recorded successfully\n\n");
}

// Display results
printf("\nVoting Results:\n");
printf("-----\n");
printf("Candidate A: %d votes (%.1f%%)\n",
       votes_A, (float)votes_A/total_votes * 100);
printf("Candidate B: %d votes (%.1f%%)\n",
       votes_B, (float)votes_B/total_votes * 100);
printf("Candidate C: %d votes (%.1f%%)\n",
       votes_C, (float)votes_C/total_votes * 100);

```

```

printf("Total votes: %d\n", total_votes);

// Determine winner
printf("\nWinner: ");
if(votes_A > votes_B && votes_A > votes_C) {
    printf("Candidate A\n");
} else if(votes_B > votes_A && votes_B > votes_C) {
    printf("Candidate B\n");
} else if(votes_C > votes_A && votes_C > votes_B) {
    printf("Candidate C\n");
} else {
    printf("Tie!\n");
}

return 0;
}

```

1. Password Validation Program

- Write a program that prompts the user to enter a password.

The password must be at least 8 characters long.

It must contain at least one uppercase letter, one lowercase letter, and one digit.

- Use `scanf` for input, `if-else` statements for validation, and a `while` loop to allow the user to try again until they enter a valid password.

```

// Exercise 1
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char password[50];
    int valid, hasUpper, hasLower, hasDigit;

    do {
        hasUpper = hasLower = hasDigit = 0;
        valid = 1;

        printf("Enter a password: ");
        scanf("%s", password);

        if (strlen(password) < 8) {
            printf("Password must be at least 8 characters long\n");
            valid = 0;
            continue;
        }

        for (int i = 0; password[i] != '\0'; i++) {
            if (isupper(password[i])) hasUpper = 1;

```

```

        if (islower(password[i])) hasLower = 1;
        if (isdigit(password[i])) hasDigit = 1;
    }

    if (!hasUpper) {
        printf("Password must contain at least one uppercase letter\n");
        valid = 0;
    }
    if (!hasLower) {
        printf("Password must contain at least one lowercase letter\n");
        valid = 0;
    }
    if (!hasDigit) {
        printf("Password must contain at least one digit\n");
        valid = 0;
    }

    } while (!valid);

    printf("Password is valid!\n");
    return 0;
}

```

2. Number Guessing Game with Feedback

- Create a number guessing game where the program generates a random number between 1 and 100.

The user tries to guess the number.

The program provides feedback: "Too high", "Too low", or "Correct!".

- Use a do while loop to allow the user to keep guessing until they guess correctly.

```

// Exercise 2
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int secret_number, guess;

    // Initialize random seed
    srand(time(NULL));
    // Generate random number between 1 and 100
    secret_number = rand() % 100 + 1;

    printf("Welcome to the Number Guessing Game!\n");
    printf("Try to guess the number between 1 and 100\n");

    do {
        printf("\nEnter your guess: ");

```

```

scanf("%d", &guess);

if (guess > secret_number) {
    printf("Too high!");
} else if (guess < secret_number) {
    printf("Too low!");
} else {
    printf("Correct! You've won!\n");
}
} while (guess != secret_number);

return 0;
}

```

3. User Login Simulation

- Write a program that simulates a user login system.

The user has up to three attempts to enter the correct username and password. Use `scanf` for input, `if-else` for checking credentials, and a `while` loop for the attempts. After three failed attempts, display "Account Locked".

```

// Exercise 3
#include <stdio.h>
#include <string.h>

int main() {
    // Predefined credentials
    const char correct_username[] = "admin";
    const char correct_password[] = "pass123";

    char username[50];
    char password[50];
    int attempts = 0;
    int max_attempts = 3;
    int login_successful = 0;

    printf("Welcome to the Login System\n");

    while (attempts < max_attempts && !login_successful) {
        printf("\nAttempt %d of %d\n", attempts + 1, max_attempts);
        printf("Username: ");
        scanf("%s", username);
        printf("Password: ");
        scanf("%s", password);

        if (strcmp(username, correct_username) == 0 &&
            strcmp(password, correct_password) == 0) {
            login_successful = 1;
            printf("\nLogin successful! Welcome, %s!\n", username);
        }
    }
}

```

```

    } else {
        printf("\nInvalid username or password.\n");
        attempts++;

        if (attempts < max_attempts) {
            printf("Attempts remaining: %d\n", max_attempts - attem
        }
    }
}

if (!login_successful) {
    printf("\n*** Account Locked ***\n");
    printf("Maximum attempts exceeded. Please contact administrator
}

return 0;
}

```

4. Simple Menu-Driven Calculator

- Develop a calculator that performs addition, subtraction, multiplication, or division based on user choice.

Display a menu for the user to choose an operation.

Use if-else if-else statements to perform the selected operation.

Use a do while loop to allow the user to perform multiple calculations until they choose to exit.

```

// Exercise 4
#include <stdio.h>

int main() {
    char operator;
    double num1, num2, result;
    int continue_calc = 1;

    printf("Simple Calculator\n");

    do {
        printf("\nAvailable Operations:\n");
        printf("+ : Addition\n");
        printf("- : Subtraction\n");
        printf("* : Multiplication\n");
        printf("/ : Division\n");
        printf("q : Quit\n");

        printf("\nEnter first number: ");
        scanf("%lf", &num1);

        printf("Enter operator (+, -, *, /, q): ");
        scanf(" %c", &operator);
    } while (operator != 'q');
}

```

```

    if (operator == 'q') {
        printf("Calculator closing...\n");
        break;
    }

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

    switch (operator) {
        case '+':
            result = num1 + num2;
            printf("%.2lf + %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '-':
            result = num1 - num2;
            printf("%.2lf - %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '*':
            result = num1 * num2;
            printf("%.2lf * %.2lf = %.2lf\n", num1, num2, result);
            break;

        case '/':
            if (num2 != 0) {
                result = num1 / num2;
                printf("%.2lf / %.2lf = %.2lf\n", num1, num2, result);
            } else {
                printf("Error: Division by zero!\n");
            }
            break;

        default:
            printf("Error: Invalid operator!\n");
    }

    printf("\nPress Enter to continue...");
    getchar();
    getchar();

} while (continue_calc);

return 0;
}

```

5. Prime Number Generator

- Write a program that asks the user for a number N and prints all prime numbers up to N.

Use while loops for iteration.
Use if statements to check for primality.

```
// Exercise 5

#include <stdio.h>

int main() {
    int n, i = 2, j;

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

    printf("Prime numbers up to %d are:\n", n);

    while (i <= n) {
        int is_prime = 1;
        j = 2;

        while (j <= i/2) {
            if (i % j == 0) {
                is_prime = 0;
                break;
            }
            j++;
        }

        if (is_prime == 1) {
            printf("%d ", i);
        }
        i++;
    }

    return 0;
}
```

6. ATM Withdrawal Simulation

- Create a program that simulates an ATM withdrawal.

The user enters a PIN code; they have three attempts to enter it correctly.
If the PIN is correct, allow the user to withdraw cash in multiples of \$10 up to their account balance.
Use if-else, scanf, and while loops.

```
// Exercise 6
#include <stdio.h>

int main() {
    const int CORRECT_PIN = 1234;
    double balance = 1000.00;
```

```

int pin, attempts = 0;
int amount;
int authenticated = 0;

printf("Welcome to the ATM\n");

while (attempts < 3 && !authenticated) {
    printf("\nEnter your PIN: ");
    scanf("%d", &pin);

    if (pin == CORRECT_PIN) {
        authenticated = 1;
        printf("\nPIN accepted!\n");
    } else {
        attempts++;
        printf("Incorrect PIN. %d attempts remaining.\n", 3 - attempts);
    }
}

if (!authenticated) {
    printf("\n*** Card blocked ***\n");
    printf("Please contact your bank.\n");
    return 0;
}

// ATM Transaction Menu
printf("\nCurrent Balance: $%.2f\n", balance);

do {
    printf("\nEnter withdrawal amount (multiple of $10): $");
    scanf("%d", &amount);

    if (amount % 10 != 0) {
        printf("Error: Amount must be a multiple of $10\n");
    } else if (amount > balance) {
        printf("Error: Insufficient funds\n");
    } else if (amount <= 0) {
        printf("Error: Invalid amount\n");
    } else {
        balance -= amount;
        printf("\nWithdrawing $%d\n", amount);
        printf("Remaining balance: $%.2f\n", balance);
        printf("Please take your cash\n");
        break;
    }
} while (1);

printf("\nThank you for using our ATM\n");
return 0;
}

```

7. Temperature Conversion with Input Validation

- Write a program that converts temperatures between Celsius and Fahrenheit.

Ask the user which conversion they want to perform.

Use if-else statements for selection.

Use a do while loop to validate the user's choice.

```
// Exercise 7
#include <stdio.h>

int main() {
    char choice;
    double temperature, converted;

    printf("Temperature Converter\n");

    do {
        printf("\nSelect conversion type:\n");
        printf("1. Celsius to Fahrenheit (C)\n");
        printf("2. Fahrenheit to Celsius (F)\n");
        printf("Q. Quit\n");
        printf("Enter choice (C/F/Q): ");
        scanf(" %c", &choice);

        if (choice == 'Q' || choice == 'q') {
            printf("Exiting program...\n");
            break;
        }

        printf("Enter temperature: ");
        scanf("%lf", &temperature);

        switch (choice) {
            case 'C':
            case 'c':
                converted = temperature * (9.0/5.0) + 32;
                printf("%.2lf°C = %.2lf°F\n", temperature, converted);
                break;

            case 'F':
            case 'f':
                converted = (temperature - 32) * (5.0/9.0);
                printf("%.2lf°F = %.2lf°C\n", temperature, converted);
                break;

            default:
                printf("Error: Invalid choice!\n");
        }

        printf("\nPress Enter to continue...");
        getchar();
        getchar();
    }
```

```
    } while (1);

    return 0;
}
```

8. Factorial Calculation with Error Handling

- Develop a program that calculates the factorial of a number entered by the user.

Ensure the number is non-negative.

Use scanf, if-else, and a while loop for the calculation.

```
// Exercise 8
#include <stdio.h>

int main() {
    int n;
    unsigned long long fact = 1;

    printf("Enter a number to calculate factorial: ");
    scanf("%d", &n);

    if (n < 0) {
        printf("Error: Factorial is not defined for negative numbers\n");
        return 1;
    }

    if (n == 0) {
        printf("Factorial of 0 = 1\n");
        return 0;
    }

    int i = 1;
    while (i <= n) {
        // Check for overflow
        if (fact > (18446744073709551615ULL / i)) {
            printf("Error: Result too large to calculate\n");
            return 1;
        }
        fact *= i;
        i++;
    }

    printf("Factorial of %d = %llu\n", n, fact);
    return 0;
}
```

9. Multiplication Table Printer

- Write a program that asks the user for a number and prints its multiplication table up to 12.

Use a while loop for iteration.

Use scanf and if to handle invalid input (e.g., non-positive numbers).

```
// Exercise 9
#include <stdio.h>

int main() {
    int number;
    int i = 1;

    printf("Enter a number for multiplication table: ");
    if (scanf("%d", &number) != 1) {
        printf("Error: Invalid input\n");
        return 1;
    }

    if (number <= 0) {
        printf("Error: Please enter a positive number\n");
        return 1;
    }

    printf("\nMultiplication table of %d:\n", number);
    printf("-----\n");

    while (i <= 12) {
        printf("%d x %d = %d\n", number, i, number * i);
        i++;
    }

    return 0;
}
```

10. Student Grade Analyzer

- Create a program that reads marks for five subjects.

Calculate the average and determine the grade:

Average	90: Grade A
Average	80: Grade B
Average	70: Grade C
Average	60: Grade D
Else:	Grade F

Use scanf, if-else if-else, and a while loop to read inputs.

```
// Exercise 10
#include <stdio.h>
```

```

int main() {
    float marks[5];
    float sum = 0, average;
    int i = 0;

    printf("Enter marks for 5 subjects\n");

    while (i < 5) {
        printf("Enter mark for subject %d (0-100): ", i + 1);
        scanf("%f", &marks[i]);

        if (marks[i] < 0 || marks[i] > 100) {
            printf("Error: Mark should be between 0 and 100\n");
            continue;
        }

        sum += marks[i];
        i++;
    }

    average = sum / 5;
    printf("\nAverage marks: %.2f\n", average);
    printf("Grade: ");

    if (average >= 90) {
        printf("A\n");
    } else if (average >= 80) {
        printf("B\n");
    } else if (average >= 70) {
        printf("C\n");
    } else if (average >= 60) {
        printf("D\n");
    } else {
        printf("F\n");
    }

    return 0;
}

```

11. Login Attempts with Do While Loop

- Write a program where the user must enter a specific code to proceed.

Use a do while loop to keep prompting until the correct code is entered.
Provide feedback if the code is incorrect.

// Exercise 11

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

```
int main() {
```

```

const int CORRECT_CODE = 5678;
int user_code;
int attempts = 0;
const int MAX_ATTEMPTS = 3;

printf("Security System\n");
printf("-----\n");

do {
    printf("\nEnter access code: ");
    scanf("%d", &user_code);
    attempts++;

    if (user_code == CORRECT_CODE) {
        printf("\nAccess granted!\n");
        break;
    } else {
        if (attempts < MAX_ATTEMPTS) {
            printf("Incorrect code. %d attempts remaining.\n",
                MAX_ATTEMPTS - attempts);

            if (user_code > CORRECT_CODE) {
                printf("Hint: Try a lower number\n");
            } else {
                printf("Hint: Try a higher number\n");
            }
        }
    }

    if (attempts >= MAX_ATTEMPTS) {
        printf("\nSystem locked! Too many incorrect attempts.\n");
        break;
    }

} while (1);

return 0;
}

```

12. Sum of Positive Numbers

- Develop a program that keeps accepting numbers from the user.

Sum only the positive numbers entered.

The program stops when the user enters zero.

Use while loop and if statements.

```

// Exercise 12
#include <stdio.h>

int main() {

```

```

double number;
double sum = 0;
int count = 0;

printf("Enter numbers (0 to stop):\n");

while (1) {
    printf("Enter a number: ");
    scanf("%lf", &number);

    if (number == 0) {
        break;
    }

    if (number > 0) {
        sum += number;
        count++;
    }
}

printf("\nResults:\n");
printf("Sum of positive numbers: %.2f\n", sum);

if (count > 0) {
    printf("Average of positive numbers: %.2f\n", sum / count);
    printf("Number of positive entries: %d\n", count);
} else {
    printf("No positive numbers were entered\n");
}

return 0;
}

```

13. Menu-Driven Shape Area Calculator

- Create a program that calculates the area of different shapes based on user choice.

Options: Circle, Rectangle, Triangle
 Use if-else if-else for menu selection.
 Use while loop to allow multiple calculations.

```

// Exercise 13
#include <stdio.h>
#include <math.h>

int main() {
    int choice;
    double area;
    const double PI = 3.14159;

    do {

```



```

printf("\nShape Area Calculator\n");
printf("-----\n");
printf("1. Circle\n");
printf("2. Rectangle\n");
printf("3. Triangle\n");
printf("4. Exit\n");
printf("\nEnter your choice (1-4): ");
scanf("%d", &choice);

switch (choice) {
    case 1: {
        double radius;
        printf("Enter radius of circle: ");
        scanf("%lf", &radius);

        if (radius < 0) {
            printf("Error: Radius cannot be negative\n");
            continue;
        }

        area = PI * radius * radius;
        printf("Area of circle = %.2lf square units\n", area);
        break;
    }

    case 2: {
        double length, width;
        printf("Enter length of rectangle: ");
        scanf("%lf", &length);
        printf("Enter width of rectangle: ");
        scanf("%lf", &width);

        if (length < 0 || width < 0) {
            printf("Error: Dimensions cannot be negative\n");
            continue;
        }

        area = length * width;
        printf("Area of rectangle = %.2lf square units\n", area);
        break;
    }

    case 3: {
        double base, height;
        printf("Enter base of triangle: ");
        scanf("%lf", &base);
        printf("Enter height of triangle: ");
        scanf("%lf", &height);

        if (base < 0 || height < 0) {
            printf("Error: Dimensions cannot be negative\n");
            continue;
        }
    }
}

```

```

        area = 0.5 * base * height;
        printf("Area of triangle = %.2lf square units\n", area)
        break;
    }

    case 4:
        printf("Thank you for using the calculator!\n");
        return 0;

    default:
        printf("Invalid choice! Please select 1-4\n");
    }

    printf("\nPress Enter to continue...");
    getchar();
    getchar();

} while (1);

return 0;
}

```

14. Palindrome Checker for Numbers

- Write a program that checks if a given number is a palindrome.

Use scanf for input, if-else for comparison, and a while loop for reversing the number.

```

// Exercise 14
#include <stdio.h>

int main() {
    int number, original, reversed = 0, remainder;

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

    if (number < 0) {
        printf("Error: Negative numbers cannot be palindromes\n");
        return 1;
    }

    original = number;

    while (number > 0) {
        remainder = number % 10;
        reversed = reversed * 10 + remainder;
        number = number / 10;
    }
}

```

```

printf("\nOriginal number: %d\n", original);
printf("Reversed number: %d\n", reversed);

if (original == reversed) {
    printf("\n%d is a palindrome!\n", original);
} else {
    printf("\n%d is not a palindrome.\n", original);
}

return 0;
}

```

15. Even or Odd Number Counter

- Develop a program that counts the number of even and odd numbers entered by the user.

The program stops when the user enters -1.
Use while loop and if statements

```

// Exercise 15
#include <stdio.h>

int main() {
    int number, original, reversed = 0, remainder;

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

    if (number < 0) {
        printf("Error: Negative numbers cannot be palindromes\n");
        return 1;
    }

    original = number;

    while (number > 0) {
        remainder = number % 10;
        reversed = reversed * 10 + remainder;
        number = number / 10;
    }

    printf("\nOriginal number: %d\n", original);
    printf("Reversed number: %d\n", reversed);

    if (original == reversed) {
        printf("\n%d is a palindrome!\n", original);
    } else {
        printf("\n%d is not a palindrome.\n", original);
    }
}

```

```
    return 0;
}
```

16. Simple Login System with Lockout

- Create a login system that locks out the user after three incorrect attempts.

Use if-else, scanf, and a while loop for attempts.

```
// Exercise 16
#include <stdio.h>
#include <string.h>

int main() {
    const char CORRECT_USERNAME[] = "admin";
    const char CORRECT_PASSWORD[] = "secure123";
    char username[50];
    char password[50];
    int attempts = 0;
    int account_locked = 0;

    printf("Secure Login System\n");
    printf("-----\n");

    while (attempts < 3) {
        printf("\nAttempt %d of 3\n", attempts + 1);
        printf("Username: ");
        scanf("%s", username);
        printf("Password: ");
        scanf("%s", password);

        if (strcmp(username, CORRECT_USERNAME) == 0 &&
            strcmp(password, CORRECT_PASSWORD) == 0) {
            printf("\nLogin successful!\n");
            printf("Welcome, %s!\n", username);
            return 0;
        } else {
            attempts++;
            if (attempts < 3) {
                printf("\nIncorrect username or password\n");
                printf("Attempts remaining: %d\n", 3 - attempts);
            }
        }
    }

    printf("\n*** ACCOUNT LOCKED ***\n");
    printf("Maximum login attempts exceeded\n");
    printf("Please contact system administrator\n");

    return 1;
}
```

```
}
```

17. Age Validator

- Write a program that asks for the user's age.

If the age is less than zero or greater than 120, prompt again.

Use scanf, if-else, and a do while loop.

```
// Exercise 17
#include <stdio.h>

int main() {
    int age;
    int valid_input = 0;

    printf("Age Validation Program\n");
    printf("-----\n");

    do {
        printf("\nEnter your age (0-120): ");

        if (scanf("%d", &age) != 1) {
            printf("Error: Please enter a valid number\n");
            while (getchar() != '\n'); // Clear input buffer
            continue;
        }

        if (age < 0) {
            printf("Error: Age cannot be negative\n");
        } else if (age > 120) {
            printf("Error: Age cannot be greater than 120\n");
        } else {
            valid_input = 1;
        }

    } while (!valid_input);

    printf("\nValid age entered: %d\n", age);

    if (age < 13) {
        printf("Category: Child\n");
    } else if (age < 20) {
        printf("Category: Teenager\n");
    } else if (age < 65) {
        printf("Category: Adult\n");
    } else {
        printf("Category: Senior\n");
    }

    return 0;
}
```

```
}
```

18. Factor Finder

- Develop a program that finds all factors of a number entered by the user.

Use while loop and if statements.

```
// Exercise 18
#include <stdio.h>

int main() {
    int number, i = 1;
    int factor_count = 0;

    printf("Enter a number to find its factors: ");
    scanf("%d", &number);

    if (number <= 0) {
        printf("Error: Please enter a positive number\n");
        return 1;
    }

    printf("\nFactors of %d are: ", number);

    while (i <= number) {
        if (number % i == 0) {
            printf("%d ", i);
            factor_count++;
        }
        i++;
    }

    printf("\nTotal number of factors: %d\n", factor_count);

    if (factor_count == 2) {
        printf("Note: %d is a prime number!\n", number);
    }

    return 0;
}
```

19. Password Strength Checker

- Create a program that checks the strength of a password entered by the user.

Criteria:

Length

Use of numbers
Use of special characters

Provide feedback on how to improve the password.
Use if-else if-else and scanf.

```
// Exercise 19
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char password[50];
    int length;
    int has_upper = 0, has_lower = 0;
    int has_digit = 0, has_special = 0;
    int strength = 0;

    printf("Enter a password: ");
    scanf("%s", password);

    length = strlen(password);

    for (int i = 0; i < length; i++) {
        if (isupper(password[i])) {
            has_upper = 1;
        } else if (islower(password[i])) {
            has_lower = 1;
        } else if (isdigit(password[i])) {
            has_digit = 1;
        } else if (strchr("!@#$$%^&*()-+", password[i])) {
            has_special = 1;
        }
    }

    printf("\nPassword Strength Analysis:\n");
    printf("-----\n");

    if (length < 8) {
        printf("- Password is too short (minimum 8 characters)\n");
    } else {
        strength++;
    }

    if (has_upper && has_lower) {
        strength++;
    } else {
        printf("- Add both uppercase and lowercase letters\n");
    }

    if (has_digit) {
        strength++;
    } else {
```

```

        printf("- Add at least one number\n");
    }

    if (has_special) {
        strength++;
    } else {
        printf("- Add at least one special character (!@#$%^&*()-+)\n")
    }

    printf("\nStrength Rating: ");
    if (strength == 4) {
        printf("Strong\n");
    } else if (strength >= 2) {
        printf("Moderate\n");
    } else {
        printf("Weak\n");
    }

    return 0;
}

```

20. Decimal to Binary Converter

- Write a program that converts a decimal number to binary.

Use while loop for the conversion.
Use scanf and if to handle invalid input.

```

// Exercise 20
#include <stdio.h>

int main() {
    int decimal, binary = 0;
    int remainder, place = 1;

    printf("Enter a decimal number: ");
    if (scanf("%d", &decimal) != 1) {
        printf("Error: Invalid input\n");
        return 1;
    }

    if (decimal < 0) {
        printf("Error: Please enter a positive number\n");
        return 1;
    }

    int original = decimal;

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

```



```

        place = place * 10;
        decimal = decimal / 2;
    }

    printf("Binary equivalent of %d is %d\n", original, binary);

    return 0;
}

```

21. Continuous Average Calculator

- Develop a program that continuously accepts numbers and calculates the average.

The user can stop entering numbers by inputting 0.
Use while loop and if statements

```

// Exercise 21

#include <stdio.h>

int main() {
    int count = 0;
    double sum = 0.0;
    double number;

    printf("Enter numbers to calculate average (enter 0 to stop):\n");

    while (1) {
        printf("Enter a number: ");
        scanf("%lf", &number);

        if (number == 0) {
            break;
        }

        sum += number;
        count++;
    }

    if (count > 0) {
        double average = sum / count;
        printf("Average: %.2lf\n", average);
    } else {
        printf("No numbers were entered.\n");
    }

    return 0;
}

```

22. Simple Voting System

- Create a program that allows users to vote for candidates A, B, or C.

Use if-else if-else for option selection.

Use a while loop to accept multiple votes.

End voting when the user inputs 'N' or 'n'.

```
// Exercise 22
#include <stdio.h>

int main() {
    int voteA = 0, voteB = 0, voteC = 0;
    char vote;
    char continue_voting;

    printf("Simple Voting System\n");
    printf("Vote for: A, B, or C\n");

    while (1) {
        printf("\nEnter your vote (A/B/C): ");
        scanf(" %c", &vote);

        if (vote == 'A' || vote == 'a') {
            voteA++;
            printf("Vote recorded for Candidate A\n");
        }
        else if (vote == 'B' || vote == 'b') {
            voteB++;
            printf("Vote recorded for Candidate B\n");
        }
        else if (vote == 'C' || vote == 'c') {
            voteC++;
            printf("Vote recorded for Candidate C\n");
        }
        else {
            printf("Invalid vote! Please vote A, B, or C only\n");
            continue;
        }

        printf("Continue voting? (Y/N): ");
        scanf(" %c", &continue_voting);

        if (continue_voting == 'N' || continue_voting == 'n') {
            break;
        }
    }

    printf("\nVoting Results:\n");
    printf("Candidate A: %d votes\n", voteA);
    printf("Candidate B: %d votes\n", voteB);
    printf("Candidate C: %d votes\n", voteC);
    printf("Total votes: %d\n", voteA + voteB + voteC);
}
```

```
    return 0;
}
```

23. Gross Pay Calculator

- Write a program that calculates the gross pay for employees.

Overtime is paid at 1.5 times the regular rate for hours worked over 40.
Use scanf, if-else, and while loop for multiple employees.

```
// Exercise 23
```

```
#include <stdio.h>

int main() {
    float hours, rate, gross_pay;
    char continue_calc;

    while (1) {
        printf("\nEnter hours worked: ");
        scanf("%f", &hours);

        printf("Enter hourly rate: $");
        scanf("%f", &rate);

        if (hours <= 40) {
            gross_pay = hours * rate;
        } else {
            gross_pay = (40 * rate) + ((hours - 40) * rate * 1.5);
        }

        printf("\nPayroll Summary:\n");
        printf("Hours Worked: %.2f\n", hours);
        printf("Hourly Rate: $%.2f\n", rate);
        printf("Gross Pay: $%.2f\n", gross_pay);

        printf("\nCalculate another employee's pay? (Y/N): ");
        scanf(" %c", &continue_calc);

        if (continue_calc == 'N' || continue_calc == 'n') {
            break;
        }
    }

    return 0;
}
```

24. Armstrong Number Checker

- Develop a program to check if a number entered by the user is an Armstrong number.

Use while loop and if statements.

// Exercise 24

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

int main() {
    int num, originalNum, remainder, digits = 0;
    double result = 0.0;

    printf("Enter a number to check if it's an Armstrong number: ");
    scanf("%d", &num);

    originalNum = num;

    // Count number of digits
    while (originalNum != 0) {
        originalNum /= 10;
        digits++;
    }

    originalNum = num;

    // Calculate sum of digits raised to power of number of digits
    while (originalNum != 0) {
        remainder = originalNum % 10;
        result += pow(remainder, digits);
        originalNum /= 10;
    }

    if ((int)result == num) {
        printf("%d is an Armstrong number.\n", num);
    } else {
        printf("%d is not an Armstrong number.\n", num);
    }

    return 0;
}
```

25. Login with Security Questions

- Create a login system that asks security questions upon failed password attempts.

Use if-else, scanf, and while loops.

Lock the account after three failed attempts.

// Exercise 25

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

```

#include <string.h>

int main() {
    // Predefined credentials
    const char correct_username[] = "admin";
    const char correct_password[] = "1234";
    const char security_answer[] = "blue";

    char username[50];
    char password[50];
    char security_response[50];
    int attempts = 0;
    int is_locked = 0;

    printf("Login System\n");

    while (attempts < 3 && !is_locked) {
        printf("\nEnter username: ");
        scanf("%s", username);
        printf("Enter password: ");
        scanf("%s", password);

        if (strcmp(username, correct_username) == 0 &&
            strcmp(password, correct_password) == 0) {
            printf("\nLogin successful! Welcome, %s!\n", username);
            break;
        } else {
            attempts++;

            if (attempts < 3) {
                printf("\nIncorrect credentials! Attempts remaining: %d", 3 - attempts);
                printf("Security Question: What is your favorite color? ");
                printf("Answer: ");
                scanf("%s", security_response);

                if (strcmp(security_response, security_answer) == 0) {
                    printf("\nSecurity answer correct! Password reset\n");
                    break;
                } else {
                    printf("Security answer incorrect!\n");
                }
            } else {
                printf("\nAccount locked! Too many failed attempts.\n");
                printf("Please contact administrator for support.\n");
                is_locked = 1;
            }
        }
    }

    return 0;
}

```

26. Traffic Light Simulation

- Write a program that simulates traffic light behavior.

Ask the user to input a color (Red, Yellow, Green).

Display the action to take.

Use if-else if-else statements and scanf.

```
// Exercise 26
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char color[10];
    char continue_sim;

    printf("Traffic Light Simulator\n");

    while (1) {
        printf("\nEnter traffic light color (Red/Yellow/Green): ");
        scanf("%s", color);

        // Convert input to lowercase for easier comparison
        for(int i = 0; color[i]; i++) {
            color[i] = tolower(color[i]);
        }

        if (strcmp(color, "red") == 0) {
            printf("\n    RED LIGHT!\n");
            printf("- STOP completely\n");
            printf("- Wait behind the line\n");
            printf("- No turning allowed\n");
        }
        else if (strcmp(color, "yellow") == 0) {
            printf("\n    YELLOW LIGHT!\n");
            printf("- Prepare to stop\n");
            printf("- Clear intersection if too close\n");
            printf("- Exercise caution\n");
        }
        else if (strcmp(color, "green") == 0) {
            printf("\n    GREEN LIGHT!\n");
            printf("- Proceed with caution\n");
            printf("- Watch for pedestrians\n");
            printf("- Yield to crossing traffic when turning\n");
        }
        else {
            printf("Invalid color! Please enter Red, Yellow, or Green o\n");
            continue;
        }

        printf("\nContinue simulation? (Y/N): ");
        scanf(" %c", &continue_sim);
    }
}
```

```

        if (continue_sim == 'N' || continue_sim == 'n') {
            printf("\nEnding traffic light simulation. Drive safely!\n");
            break;
        }
    }

    return 0;
}

```

27. Simple Interest Calculator with Loop

- Develop a program that calculates simple interest for multiple entries.

Use `while` loop to continue calculations until the user decides to stop.
Use `if` statements to validate input.

```

// Exercise 27
#include <stdio.h>

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

    printf("Simple Interest Calculator\n");

    while (1) {
        printf("\nEnter principal amount: $");
        scanf("%f", &principal);

        printf("Enter interest rate (%%): ");
        scanf("%f", &rate);

        printf("Enter time (in years): ");
        scanf("%f", &time);

        if (principal <= 0) {
            printf("Error: Principal amount must be positive\n");
            continue;
        }

        if (rate <= 0) {
            printf("Error: Interest rate must be positive\n");
            continue;
        }

        if (time <= 0) {
            printf("Error: Time period must be positive\n");
            continue;
        }
    }
}

```

```

    interest = (principal * rate * time) / 100;

    printf("\nCalculation Results:\n");
    printf("Principal Amount: $%.2f\n", principal);
    printf("Interest Rate: %.2f%%\n", rate);
    printf("Time Period: %.2f years\n", time);
    printf("Simple Interest: $%.2f\n", interest);
    printf("Total Amount: $%.2f\n", principal + interest);

    printf("\nCalculate another interest? (Y/N): ");
    scanf(" %c", &continue_calc);

    if (continue_calc == 'N' || continue_calc == 'n') {
        printf("\nThank you for using the Simple Interest Calculator\n");
        break;
    }
}

return 0;
}

```

28. Leap Year Finder

- Create a program that finds all leap years between two years entered by the user.

Use scanf, if-else, and a while loop.

```

// Exercise 28
#include <stdio.h>

int main() {
    int start_year, end_year;
    int count = 0;

    printf("Leap Year Finder\n\n");

    printf("Enter start year: ");
    scanf("%d", &start_year);

    printf("Enter end year: ");
    scanf("%d", &end_year);

    if (start_year > end_year) {
        printf("Error: Start year should be less than end year!\n");
        return 1;
    }

    printf("\nLeap Years between %d and %d:\n", start_year, end_year);
    printf("-----\n");

    int year = start_year;

```



```

while (year <= end_year) {
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        printf("%d", year);
        count++;

        // Format output in rows of 5
        if (count % 5 == 0) {
            printf("\n");
        } else {
            printf("\t");
        }
    }
    year++;
}

if (count == 0) {
    printf("No leap years found in this range!\n");
} else {
    if (count % 5 != 0) {
        printf("\n");
    }
    printf("-----\n");
    printf("Total leap years found: %d\n", count);
}

return 0;
}

```

29. Password Masking (No Actual Masking, Just Concept)

- Write a program that asks for a password and confirms it.

Ensure both entries match.

Use scanf, if-else, and do while loop.

// Exercise 29

30. Shopping Cart Total Calculator

- Develop a program that lets the user add item prices to a shopping cart
 - Use `while` loop to accept multiple prices.
 - Apply discounts based on total amount using `if-else`.

30. Shopping Cart Total Calculator

- Develop a program that lets the user add item prices to a shopping cart.

Use while loop to accept multiple prices.

Apply discounts based on total amount using if-else.

```

// Exercise 30
#include <stdio.h>

int main() {
    float price, total = 0.0;
    int items = 0;
    char continue_shopping;

    printf("Shopping Cart Calculator\n");
    printf("-----\n");

    while (1) {
        printf("\nEnter item price: $");
        scanf("%f", &price);

        if (price <= 0) {
            printf("Invalid price! Please enter a positive amount.\n");
            continue;
        }

        total += price;
        items++;

        printf("Item added: $%.2f\n", price);
        printf("Current total: $%.2f\n", total);

        printf("\nAdd another item? (Y/N): ");
        scanf(" %c", &continue_shopping);

        if (continue_shopping == 'N' || continue_shopping == 'n') {
            break;
        }
    }

    printf("\nShopping Cart Summary\n");
    printf("-----\n");
    printf("Number of items: %d\n", items);
    printf("Subtotal: $%.2f\n", total);

    float discount = 0.0;
    if (total >= 200) {
        discount = total * 0.15;
        printf("15%% Discount applied: -$.2f\n", discount);
    } else if (total >= 100) {
        discount = total * 0.10;
        printf("10%% Discount applied: -$.2f\n", discount);
    } else if (total >= 50) {
        discount = total * 0.05;
        printf("5%% Discount applied: -$.2f\n", discount);
    }

    float final_total = total - discount;
    printf("Final Total: $%.2f\n", final_total);
}

```

```
    return 0;
}
```

31. Guess the Number within Limited Attempts

- Create a number guessing game where the user has a limited number of attempts.

Use scanf, if-else, and a while loop.

Inform the user when they have no attempts left.

```
// Exercise 31
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int targetNumber, userGuess, attempts = 0;
    const int MAX_ATTEMPTS = 5;

    srand(time(NULL));
    targetNumber = rand() % 20 + 1;

    printf("Guess the number between 1 and 20!\n");

    while (attempts < MAX_ATTEMPTS) {
        printf("You have %d attempts left.\n", MAX_ATTEMPTS - attempts);
        printf("Enter your guess: ");
        scanf("%d", &userGuess);
        attempts++;

        if (userGuess == targetNumber) {
            printf("Congratulations! You guessed the correct number in\n");
            return 0;
        } else if (userGuess < targetNumber) {
            printf("Too low!\n");
        } else {
            printf("Too high!\n");
        }
    }

    printf("Game Over! The number was %d\n", targetNumber);
    return 0;
}
```

32. Palindrome String Checker

- Write a program that checks if a string entered by the user is a palindrome.

Use scanf, if-else, and a while loop.

```
// Exercise 32
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char str[100];
    int i, j, isPalindrome = 1;

    printf("Enter a string: ");
    scanf("%s", str);

    // Convert string to lowercase
    for(i = 0; str[i]; i++) {
        str[i] = tolower(str[i]);
    }

    // Check palindrome
    j = strlen(str) - 1;
    i = 0;

    while(i < j) {
        if(str[i] != str[j]) {
            isPalindrome = 0;
            break;
        }
        i++;
        j--;
    }

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

    return 0;
}
```

33. BMI Calculator with Advice

- Develop a program that calculates the BMI and gives health advice.

Use if-else if-else for BMI categories.
Use while loop to calculate for multiple users.

```
// Exercise 33

#include <stdio.h>
```

```

int main() {
    float weight, height, bmi;
    char continue_calc;

    do {
        // Input
        printf("\nBMI Calculator\n");
        printf("Enter weight (kg): ");
        scanf("%f", &weight);

        printf("Enter height (m): ");
        scanf("%f", &height);

        // Calculate BMI
        bmi = weight / (height * height);

        // Display BMI
        printf("\nYour BMI is: %.1f\n", bmi);

        // Categorize and provide advice
        if (bmi < 18.5) {
            printf("Category: Underweight\n");
            printf("Health Advice: Consider increasing caloric intake w
        }
        else if (bmi >= 18.5 && bmi < 25) {
            printf("Category: Normal Weight\n");
            printf("Health Advice: Maintain your healthy lifestyle with
        }
        else if (bmi >= 25 && bmi < 30) {
            printf("Category: Overweight\n");
            printf("Health Advice: Focus on portion control and increas
        }
        else {
            printf("Category: Obese\n");
            printf("Health Advice: Consult a healthcare provider for a
        }

        // Ask to continue
        printf("\nCalculate another BMI? (y/n): ");
        scanf(" %c", &continue_calc);

    } while (continue_calc == 'y' || continue_calc == 'Y');

    return 0;
}

```

34. Number Pattern Printer

- Create a program that prints a number pattern based on the user's input.

Use nested while loops.
Use scanf and if to handle input.

// Exercise 34

```
#include <stdio.h>

int main() {
    int rows, pattern_choice, i, j;

    printf("Number Pattern Printer\n");
    printf("1. Increasing Triangle\n");
    printf("2. Number Square\n");
    printf("3. Number Pyramid\n");
    printf("Choose pattern (1-3): ");
    scanf("%d", &pattern_choice);

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

    if (pattern_choice == 1) {
        // Increasing Triangle Pattern
        i = 1;
        while (i <= rows) {
            j = 1;
            while (j <= i) {
                printf("%d ", j);
                j++;
            }
            printf("\n");
            i++;
        }
    }
    else if (pattern_choice == 2) {
        // Number Square Pattern
        i = 1;
        while (i <= rows) {
            j = 1;
            while (j <= rows) {
                printf("%d ", i);
                j++;
            }
            printf("\n");
            i++;
        }
    }
    else if (pattern_choice == 3) {
        // Number Pyramid Pattern
        i = 1;
        while (i <= rows) {
            // Print spaces
            j = 1;
```

```

        while (j <= rows - i) {
            printf(" ");
            j++;
        }

        // Print numbers
        j = 1;
        while (j <= 2 * i - 1) {
            printf("%d", i);
            j++;
        }
        printf("\n");
        i++;
    }
}
else {
    printf("Invalid pattern choice!\n");
}

return 0;
}

```

35. Character Frequency Counter

- Write a program that counts the frequency of each character in a string.

Use while loop and if statements.
Use scanf for input.

```

// Exercise 35
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char str[100];
    int freq[128] = {0}; // Array to store frequency of ASCII characters
    int i = 0;

    printf("Enter a string: ");
    scanf("%[^\n]s", str); // Reads entire line including spaces

    // Count frequency of each character
    while(str[i] != '\0') {
        if(isprint(str[i])) { // Only count printable characters
            freq[str[i]]++;
        }
        i++;
    }

    // Print character frequencies

```

```

printf("\nCharacter frequencies:\n");
i = 0;
while(i < 128) {
    if(freq[i] > 0 && isprint(i)) {
        if(i == ' ') {
            printf("Space: %d\n", freq[i]);
        } else {
            printf("'%c': %d\n", i, freq[i]);
        }
    }
    i++;
}

return 0;
}

```

36. Simple Bank Account Simulator

- Develop a program that simulates bank account operations:

Deposit

Withdraw

Check Balance

Use if-else if-else, scanf, and a while loop for the menu.

```

// Exercise 36
#include <stdio.h>

int main() {
    double balance = 0.0;
    int choice;
    double amount;
    int running = 1;

    printf("Welcome to Simple Bank Account Simulator\n");

    while(running) {
        printf("\nBank Menu:\n");
        printf("1. Deposit\n");
        printf("2. Withdraw\n");
        printf("3. Check Balance\n");
        printf("4. Exit\n");
        printf("Enter choice (1-4): ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                printf("Enter deposit amount: $");
                scanf("%lf", &amount);
                if(amount > 0) {
                    balance += amount;
                }
            case 2:
            case 3:
            case 4:
                // Additional logic for cases 2, 3, and 4 would go here
                break;
        }
    }
}

```



```

        printf("Successfully deposited $%.2f\n", amount);
    } else {
        printf("Invalid amount. Please enter a positive val
    }
    break;

case 2:
    printf("Enter withdrawal amount: $");
    scanf("%lf", &amount);
    if(amount > 0) {
        if(amount <= balance) {
            balance -= amount;
            printf("Successfully withdrew $%.2f\n", amount)
        } else {
            printf("Insufficient funds!\n");
        }
    } else {
        printf("Invalid amount. Please enter a positive val
    }
    break;

case 3:
    printf("Current balance: $%.2f\n", balance);
    break;

case 4:
    printf("Thank you for using our banking system!\n");
    running = 0;
    break;

default:
    printf("Invalid choice. Please select 1-4.\n");
}

return 0;
}

```

37. Guess the Word Game

- Create a game where the user guesses letters to form a word.

Provide feedback on correct and incorrect guesses.
Use while loop and if statements.

```

// Exercise 37
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {

```

```

const char word[] = "PROGRAMMING";
char guessed[50] = {0};
char display[50] = {0};
int attempts = 6;
int correctGuesses = 0;
int wordLength = strlen(word);
int alreadyGuessed = 0;
char guess;

// Initialize display with underscores
for(int i = 0; i < wordLength; i++) {
    display[i] = '_';
}
display[wordLength] = '\0';

printf("Welcome to Guess the Word!\n");
printf("The word has %d letters.\n", wordLength);

while(attempts > 0 && correctGuesses < wordLength) {
    printf("\nWord: %s\n", display);
    printf("Attempts left: %d\n", attempts);
    printf("Enter a letter: ");
    scanf(" %c", &guess);
    guess = toupper(guess);

    // Check if letter was already guessed
    alreadyGuessed = 0;
    for(int i = 0; guessed[i] != '\0'; i++) {
        if(guessed[i] == guess) {
            printf("You already guessed that letter!\n");
            alreadyGuessed = 1;
            break;
        }
    }

    if(!alreadyGuessed) {
        // Add to guessed letters
        guessed[strlen(guessed)] = guess;

        // Check if guess is correct
        int foundMatch = 0;
        for(int i = 0; i < wordLength; i++) {
            if(word[i] == guess && display[i] == '_') {
                display[i] = guess;
                correctGuesses++;
                foundMatch = 1;
            }
        }

        if(!foundMatch) {
            attempts--;
            printf("Incorrect guess!\n");
        } else {

```

```

        printf("Correct guess!\n");
    }
}

// Game end
if(correctGuesses == wordLength) {
    printf("\nCongratulations! You won!\n");
    printf("The word was: %s\n", word);
} else {
    printf("\nGame Over! You ran out of attempts.\n");
    printf("The word was: %s\n", word);
}

return 0;
}

```

38. Sum of Digits Calculator

- Write a program that calculates the sum of the digits of a number entered by the user.

Use while loop and if statements.

```

// Exercise 38
#include <stdio.h>
#include <stdlib.h>

int main() {
    long long number;
    int sum = 0;
    int digit;
    long long temp;

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

    // Handle negative numbers
    if(number < 0) {
        number = -number;
        printf("Using absolute value: %lld\n", number);
    }

    // Store original number for display
    temp = number;

    // Calculate sum of digits
    while(number > 0) {
        digit = number % 10;
        sum += digit;
        number /= 10;
    }
}

```

```

printf("Sum of digits in %lld is: %d\n", temp, sum);

// Display calculation process
printf("\nCalculation breakdown:\n");
while(temp > 0) {
    digit = temp % 10;
    printf("%d", digit);
    temp /= 10;
    if(temp > 0) {
        printf(" + ");
    }
}
printf(" = %d\n", sum);

return 0;
}

```

39. Number Sequence Validator

- Develop a program that checks if a sequence of numbers entered by the user is increasing, decreasing, or neither.

Use scanf, if-else, and while loop.

```

// Exercise 39

#include <stdio.h>

int main() {
    int current, previous;
    int count = 0;
    int increasing = 1;
    int decreasing = 1;

    printf("Enter a sequence of numbers (enter -999 to stop):\n");

    // Read first number
    scanf("%d", &previous);
    count++;

    while(1) {
        scanf("%d", &current);

        // Check for end of sequence
        if(current == -999) {
            break;
        }

        // Compare with previous number
        if(current > previous) {

```

```

        decreasing = 0;
    } else if(current < previous) {
        increasing = 0;
    } else {
        increasing = 0;
        decreasing = 0;
    }

    previous = current;
    count++;
}

// Analyze sequence pattern
printf("\nSequence analysis:\n");

if(count < 2) {
    printf("Need at least two numbers to determine a pattern.\n");
} else if(increasing) {
    printf("The sequence is strictly increasing.\n");
} else if(decreasing) {
    printf("The sequence is strictly decreasing.\n");
} else {
    printf("The sequence is neither strictly increasing nor decreasing.\n");
}

printf("Number of elements: %d\n", count - 1); // Subtract sentinel

return 0;
}

```

40. Prime Factors Finder

- Create a program that finds all prime factors of a number entered by the user.

Use while loops and if statements.

```

#include <stdio.h>

int main() {
    int number, originalNumber;
    int factor = 2;

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

    if(number <= 1) {
        printf("Please enter a number greater than 1.\n");
        return 1;
    }

    originalNumber = number;

```

```

printf("Prime factors of %d are: ", originalNumber);

// Find prime factors
while(number > 1) {
    if(number % factor == 0) {
        printf("%d", factor);
        number = number / factor;
        if(number > 1) {
            printf(" × ");
        }
    } else {
        factor++;
    }
}

// Calculate product verification
printf("\nVerification: ");
factor = 2;
number = originalNumber;

while(number > 1) {
    if(number % factor == 0) {
        printf("%d", factor);
        number = number / factor;
        if(number > 1) {
            printf(" × ");
        }
    } else {
        factor++;
    }
}
printf(" = %d\n", originalNumber);

return 0;
}

```

How It Works

Prime Factorization Process

- Starts with smallest prime number (2)
- Repeatedly divides by current factor when possible
- Increments factor when division not possible
- Continues until number becomes 1

Algorithm Features

- Handles any positive integer greater than 1
- Automatically finds prime factors in ascending order
- Shows multiplication expression
- Includes verification of factorization

For example:

Input: 84

Output: Prime factors of 84 are: $2 \times 2 \times 3 \times 7$

Verification: $2 \times 2 \times 3 \times 7 = 84$

Input: 100

Output: Prime factors of 100 are: $2 \times 2 \times 5 \times 5$

Verification: $2 \times 2 \times 5 \times 5 = 100$

The program provides both the prime factorization and a verification showing that the factors multiply to give the original number, making it educational and useful for understanding number composition.

Github

```
// Προγραμμα που να ορίζει ακτεια ενός κυκλου r = 10.0
// να υπολογίζει και να εμφιζει την περιφερια (2πr) και το εμβαδον του
// η τιμη το π να δηλωθει σταθερα π=3.14159
```

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

```
#define PI 3.14159
```

```
int main() {
    double r = 10.0;
    double circumference = 2 * PI * r;
    double area = PI * r * r;
```

```
    printf("Η περιφέρεια του κύκλου είναι: %.3f\n", circumference);
    printf("Το εμβαδόν του κύκλου είναι: %.2f\n", area);
```

```
    // 4 μεταβλητες int με τιμες 13, 48, 56, 24. Στην συνεχεια να υπολογισ
    int a = 13, b = 48, c = 56, d = 24;
    double average = (a + b + c + d) / 4.0;
```

```
    printf("Ο μέσος όρος των αριθμών είναι: %.2f\n", average);
```

```
    return 0;
}
```

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

```
// 1.Function to calculate sum and average of 5 float values
void calculate_sum_and_average() {
```

```
    float values[5];
    float sum = 0.0;
    float average;
```

```
    printf("Enter 5 float values:\n");
```

```

    for (int i = 0; i < 5; i++) {
        scanf("%f", &values[i]);
        sum += values[i];
    }

    average = sum / 5;

    printf("Sum: %.2f\n", sum);
    printf("Average: %.2f\n", average);
}

// 2.Convert Fahrenheit to Celsius

// Function to convert Fahrenheit to Celsius
float fahrenheit_to_celsius(float fahrenheit) {
    return (fahrenheit - 32) * 5.0 / 9.0;
}

// Function to get temperature in Fahrenheit and convert to Celsius
void convert_temperature() {
    float fahrenheit, celsius;

    printf("Enter temperature in Fahrenheit: ");
    scanf("%f", &fahrenheit);

    celsius = fahrenheit_to_celsius(fahrenheit);

    printf("Temperature in Celsius: %.2f\n", celsius);
}

// 3.Will tke an hour in the following form 16:30:12 and will convert i
// Function to convert time in HH:MM:SS format to seconds
int time_to_seconds(int hours, int minutes, int seconds) {
    return hours * 3600 + minutes * 60 + seconds;
}

// Function to get time in HH:MM:SS format and convert to seconds
void convert_time_to_seconds() {
    int hours, minutes, seconds;

    printf("Enter time in Hours:Minutes:Seconds format: ");
    scanf("%d:%d:%d", &hours, &minutes, &seconds);

    int total_seconds = time_to_seconds(hours, minutes, seconds);

    printf("Total time in seconds: %d\n", total_seconds);
}

// 4. Takes a 3 digit int and returns the digits split and the summed d
void split_and_sum_digits(int number) {
    int digit1, digit2, digit3;
    int sum;

```



```

    digit1 = number / 100;
    digit2 = (number / 10) % 10;
    digit3 = number % 10;

    sum = digit1 + digit2 + digit3;

    printf("Digit 1: %d\n", digit1);
    printf("Digit 2: %d\n", digit2);
    printf("Digit 3: %d\n", digit3);
    printf("Sum of digits: %d\n", sum);
}

// 5. takes 2 hours in the following form 16:30:12 and returns the difference
void calculate_time_difference() {
    int hours1, minutes1, seconds1;
    int hours2, minutes2, seconds2;
    int total_seconds1, total_seconds2;
    int difference_seconds;
    int difference_hours, difference_minutes, difference_seconds_final;

    printf("Enter time 1 in Hours:Minutes:Seconds format: ");
    scanf("%d:%d:%d", &hours1, &minutes1, &seconds1);

    printf("Enter time 2 in Hours:Minutes:Seconds format: ");
    scanf("%d:%d:%d", &hours2, &minutes2, &seconds2);

    total_seconds1 = time_to_seconds(hours1, minutes1, seconds1);
    total_seconds2 = time_to_seconds(hours2, minutes2, seconds2);

    difference_seconds = total_seconds2 - total_seconds1;

    difference_hours = difference_seconds / 3600;
    difference_seconds %= 3600;
    difference_minutes = difference_seconds / 60;
    difference_seconds %= 60;
    difference_seconds_final = difference_seconds;

    printf("Time difference: %d:%d:%d\n", difference_hours, difference_
}

// 6. reads the num of university students that passed and failed the f
void calculate_pass_fail_percentage() {
    int passed, failed;
    int total;
    float pass_percentage, fail_percentage;

    printf("Enter the number of students who passed: ");
    scanf("%d", &passed);
    printf("Enter the number of students who failed: ");
    scanf("%d", &failed);

    total = passed + failed;
    if (total == 0) {

```

```

        printf("No students appeared for the exam.\n");
        return;
    }

    pass_percentage = (passed / (float)total) * 100;
    fail_percentage = (failed / (float)total) * 100;

    printf("Pass percentage: %.2f%%\n", pass_percentage);
    printf("Fail percentage: %.2f%%\n", fail_percentage);
}

// 7. will read 2 integers and will return the max and min integer

void calculate_max_min() {
    int max, min;
    int num1, num2;
    printf("Enter two integers: ");
    scanf(" %d", &num1);
    scanf(" %d", &num2);
    if (num1 > num2) {
        max = num1;
        min = num2;
    } else {
        max = num2;
        min = num1;
    }

    printf("Max: %d\n", max);
    printf("Min: %d\n", min);
}

// Main function
int main() {
    int function_number;    // Variable to store the function number

    while (1) {
        printf("Enter a function number (1-8) or 0 to exit: ");
        printf("1. Calculate sum and average of 5 float values\n");
        printf("2. Convert Fahrenheit to Celsius\n");
        printf("3. Convert time in HH:MM:SS format to seconds\n");
        printf("4. Split and sum digits of a 3-digit number\n");
        printf("5. Calculate time difference between two times\n");
        printf("6. Calculate pass and fail percentage of students\n");
        printf("7. Calculate max and min of two integers\n");
        printf("8. (coming soon)\n");
        printf("Enter your choice here: ");
        scanf("%d", &function_number);

        if (function_number == 0) {
            break;
        } else if (function_number == 1) {
            calculate_sum_and_average();
        } else if (function_number == 2) {
            convert_temperature();
        }
    }
}

```

```

        } else if (function_number == 3) {
            convert_time_to_seconds();
        } else if (function_number == 4) {
            int number;
            printf("Enter a 3 digit number: ");
            scanf("%d", &number);
            split_and_sum_digits(number);
        } else if (function_number == 5) {
            calculate_time_difference();
        } else if (function_number == 6) {
            calculate_pass_fail_percentage();
        } else if (function_number == 7) {
            calculate_max_min();
        } else if (function_number == 8) {
            // Add your function here
        } else {
            printf("Invalid function number\n");
        }
    }

    return 0;
}

//Calculator

#include <stdio.h>

// Function to perform addition
float add(float a, float b) {
    return a + b;
}

// Function to perform subtraction
float subtract(float a, float b) {
    return a - b;
}

// Function to perform multiplication
float multiply(float a, float b) {
    return a * b;
}

// Function to perform division
float divide(float a, float b) {
    if (b == 0) {
        printf("Error: Division by zero!\n");
        return 0; // Or handle the error in a more robust way
    }
    return a / b;
}

int main() {
    float num1, num2, result;

```

```

char operator;

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

printf("Enter operator (+, -, *, /): ");
scanf(" %c", &operator); // The space before %c consumes any leftover

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

switch (operator) {
    case '+':
        result = add(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '-':
        result = subtract(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '*':
        result = multiply(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '/':
        result = divide(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    default:
        printf("Error: Invalid operator!\n");
}

return 0;
}

```