Control Flow Exercise

Tags	
Status	Not started

1. Find the Greatest Number Among Three Given Numbers
2. Determine if a Number is Positive or Negative
3. Determine if a Character is a Vowel or a Consonant
4. A Character Is an Alphabet or Not
5. Uppercase, Lowercase, Special Character, or Digit
6. The Number Is Positive or Negative
7. The Number Is Even or Odd
8. Greatest of Two Numbers
9. Greatest Among Three Numbers
10. Leap Year
11. The Date Is Correct or Not
12. Voting Eligibility Checker
13. Find the maximum between two numbers
14. Find the maximum between the three numbers
15. Check whether a number is negative, positive or zero
16. Check whether a number is divisible by 5 and 11 or not
17. Find whether a number is even or odd
18. Check whether a year is a leap year or not.
19. Check whether a character is an alphabet or not
20. Input any alphabet and check whether it is vowel or consonant
21. Input any character and check whether it is the alphabet, digit or special character
22. Check whether a character is an uppercase or lowercase alphabet
23. Input week number and print weekday.
24. Input month number and print number of days in that month.
25. Count the total number of notes in a given amount
26. Input angles of a triangle and check whether the triangle is valid or not.
27. Input all sides of a triangle and check whether the triangle is valid or not.
28. Check whether the triangle is an equilateral, isosceles or scalene triangle.
29. Find all roots of a quadratic equation
30. Calculate profit or loss.

1. Find the Greatest Number Among Three Given Numbers

Write a program in the C programming language to determine the largest number among three integers entered by the user.

Detailed Requirements:

1. The user will input three integer numbers.
2. The program must identify and display the largest number among the three.
3. Ensure the program handles all possible cases correctly, including:
   - All three numbers are the same.
   - Two of the three numbers are equal and are the largest.
4. Display the result in the format:
   - `"The greatest number is: <value>"`.

Example:

- **Input:**

  ```
  Enter the first number: 12
  Enter the second number: 25
  Enter the third number: 7
  ```

  Output:

  ```
  The greatest number is: 25
  ```

- **Input:**

  ```
  Enter the first number: -5
  Enter the second number: -10
  Enter the third number: -3
  ```

  Output:

  ```
  The greatest number is: -3
  ```

Hints:

- Use conditional statements such as `if-else` or the ternary operator (`?:`) to compare values.

- Ensure the program is efficient, readable, and handles edge cases appropriately.

2. Determine if a Number is Positive or Negative

Write a program in the C programming language to determine whether a given number is positive, negative, or zero.

Detailed Requirements:

1. The user will input a single integer.
2. The program must check and display:
   - If the number is positive, display: `"The number is positive."`
   - If the number is negative, display: `"The number is negative."`
   - If the number is zero, display: `"The number is zero."`

Example:

- **Input:**

  `Enter a number: 10`

  Output:

  `The number is positive.`
- **Input:**

  `Enter a number: -25`

  Output:

  `The number is negative.`
- **Input:**

  `Enter a number: 0`

  Output:

  `The number is zero.`

Hints:

- Use conditional statements (`if`, `else if`, `else`) to evaluate the number.
- Consider edge cases such as 0 and negative values.

- Ensure the program has clear prompts and outputs.

3. Determine if a Character is a Vowel or a Consonant

Write a program in the C programming language to check whether a given character is a vowel or a consonant.

Detailed Requirements:

1. The user will input a single alphabet character.
2. The program must determine:
   - If the character is a vowel (`a, e, i, o, u` or `A, E, I, O, U`), display: `"The character is a vowel."`
   - If the character is a consonant, display: `"The character is a consonant."`
3. If the input is not an alphabet character, display: `"Invalid input. Please enter an alphabet."`

Example:

- **Input:**

  `Enter a character: A`

  Output:

  `The character is a vowel.`

- **Input:**

  `Enter a character: x`

  Output:

  `The character is a consonant.`

- **Input:**

  `Enter a character: 7`

  Output:

  `Invalid input. Please enter an alphabet.`

4. A Character Is an Alphabet or Not

Write a program in the C programming language to check whether a given character is an alphabet or not.

Detailed Requirements:

1. The user will input a single character.
2. The program must determine:
   - If the character is an alphabet (either uppercase `A-Z` or lowercase `a-z`), display: `"The character is an alphabet."`
   - Otherwise, display: `"The character is not an alphabet."`

Example:

- **Input:**

  `Enter a character: A`

  Output:

  `The character is an alphabet.`

- **Input:**

  `Enter a character: z`

  Output:

  `The character is an alphabet.`

- **Input:**

  `Enter a character: 9`

  Output:

  `The character is not an alphabet.`

- **Input:**

  `Enter a character: @`

  Output:

  `The character is not an alphabet.`

Hints:

- Use the ASCII values to check if the character is within the ranges of `A-Z` (65-90) or `a-z` (97-122).
- Use conditional statements (`if-else`) for comparisons.
- Ensure the program provides clear prompts and outputs.

5. Uppercase, Lowercase, Special Character, or Digit

Write a program in the C programming language to classify a given character as one of the following:

- Uppercase letter
- Lowercase letter
- Digit
- Special character

Detailed Requirements:

1. The user will input a single character.
2. The program must determine:
    - If the character is an uppercase letter (`A-Z`), display: `"The character is an uppercase letter."`
    - If the character is a lowercase letter (`a-z`), display: `"The character is a lowercase letter."`
    - If the character is a digit (`0-9`), display: `"The character is a digit."`
    - Otherwise, display: `"The character is a special character."`

Example:

- **Input:**

  `Enter a character: A`

  Output:

  `The character is an uppercase letter.`

- **Input:**

  `Enter a character: m`

  Output:

  `The character is a lowercase letter.`

- **Input:**

  `Enter a character: 5`

  Output:

  `The character is a digit.`

- **Input:**

  `Enter a character: @`

  Output:

  `The character is a special character.`

Hints:

- Use the ASCII values to check ranges:
  - Uppercase letters: 65–90 (`A-Z`)
  - Lowercase letters: 97–122 (`a-z`)
  - Digits: 48–57 (`0-9`)
- Use conditional statements (`if-else`) for classification.
- Ensure that the program outputs clear and concise messages for each category.

6. The Number Is Positive or Negative

Write a program in the C programming language to determine whether a given number is positive, negative, or zero.

Detailed Requirements:

1. The user will input an integer.
2. The program must determine:
   - If the number is greater than zero, display: `"The number is positive."`
   - If the number is less than zero, display: `"The number is negative."`
   - If the number is equal to zero, display: `"The number is zero."`

Example:

- **Input:**

  `Enter a number: 15`

  Output:

  `The number is positive.`

- **Input:**

  `Enter a number: -8`

Output:

```
The number is negative.
```

- **Input:**

  ```
  Enter a number: 0
  ```

  Output:

  ```
  The number is zero.
  ```

Hints:

- Use conditional statements (`if-else`) to evaluate the number.
- Ensure proper prompts for the user to enter the number and clear output messages for the results.

7. The Number Is Even or Odd

Write a program in the C programming language to determine whether a given integer is even or odd.

Detailed Requirements:

1. The user will input an integer.
2. The program must determine:
   - If the number is divisible by 2 without a remainder, display: `"The number is even."`
   - Otherwise, display: `"The number is odd."`

Example:

- **Input:**

  ```
  Enter a number: 10
  ```

  Output:

  ```
  The number is even.
  ```

- **Input:**

  ```
  Enter a number: 7
  ```

  Output:

  ```
  The number is odd.
  ```

Hints:

- Use the modulus operator (`%`) to check the remainder when dividing the number by 2.
- Ensure the program handles both positive and negative numbers.
- Provide clear prompts and outputs to guide the user.

8. Greatest of Two Numbers

Write a program in the C programming language to determine the larger of two numbers entered by the user.

Detailed Requirements:

1. The user will input two integers.
2. The program must determine:
   - If the first number is greater than the second, display: `"The greatest number is: <first number>."`
   - If the second number is greater than the first, display: `"The greatest number is: <second number>."`
   - If both numbers are equal, display: `"Both numbers are equal."`

Example:

- **Input:**

  ```
  Enter the first number: 8
  Enter the second number: 15
  ```

  Output:

  ```
  The greatest number is: 15.
  ```

- **Input:**

  ```
  Enter the first number: 20
  Enter the second number: 10
  ```

  Output:

  ```
  The greatest number is: 20.
  ```

- **Input:**

  ```
  Enter the first number: 12
  Enter the second number: 12
  ```

  Output:

```
Both numbers are equal.
```

Hints:

- Use conditional statements (`if-else`) to compare the two numbers.
- Ensure proper handling of equality and include clear prompts and outputs.

9. Greatest Among Three Numbers

Write a program in the C programming language to determine the greatest of three numbers entered by the user.

Detailed Requirements:

1. The user will input three integers.
2. The program must determine and display the greatest number among the three.
3. If two or more numbers are equal and are the largest, display the largest value once.
4. Provide clear output for all cases.

Example:

- **Input:**

  ```
  Enter the first number: 10
  Enter the second number: 25
  Enter the third number: 7
  ```

  Output:

  ```
  The greatest number is: 25.
  ```

- **Input:**

  ```
  Enter the first number: 5
  Enter the second number: 5
  Enter the third number: 5
  ```

  Output:

  ```
  The greatest number is: 5.
  ```

- **Input:**

  ```
  Enter the first number: -3
  Enter the second number: -1
  ```

`Enter the third number: -2`

Output:

`The greatest number is: -1.`

Hints:

- Use nested `if-else` statements or logical operators to compare the three numbers.
- Ensure proper handling of equality between numbers.
- Provide clear prompts and outputs for user interaction.

10. Leap Year

Write a program in the C programming language to determine whether a given year is a leap year or not.

Detailed Requirements:

1. The user will input a year (positive integer).
2. The program must determine if the year is a leap year using the following rules:
   - A year is a leap year if:
     - It is divisible by 4 and not divisible by 100, **or**
     - It is divisible by 400.
3. If the year is a leap year, display: `"The year <year> is a leap year."`
4. Otherwise, display: `"The year <year> is not a leap year."`

Example:

- **Input:**

  `Enter a year: 2024`

  Output:

  `The year 2024 is a leap year.`

- **Input:**

  `Enter a year: 1900`

  Output:

`The year 1900 is not a leap year.`

- **Input:**

  `Enter a year: 2000`

  Output:

  `The year 2000 is a leap year.`

Hints:

- Use the modulus operator (`%`) to check divisibility.
- Implement the conditions:
  - `(year % 4 == 0 && year % 100 != 0)` OR `(year % 400 == 0)`.
- Ensure the program handles invalid inputs (e.g., non-positive years) with an appropriate message.

11. The Date Is Correct or Not

Write a program in the C programming language to validate whether a given date (day, month, year) is correct or not.

Detailed Requirements:

1. The user will input three integers: day, month, and year.
2. The program must validate:
   - The `year` is a positive number (e.g., greater than 0).
   - The `month` is between 1 and 12.
   - The `day` is valid for the given month and year, considering:
     - Months with 31 days: January (1), March (3), May (5), July (7), August (8), October (10), December (12).
     - Months with 30 days: April (4), June (6), September (9), November (11).
     - February (2): 28 days in a common year and 29 days in a leap year.
   - A year is a leap year if:
     - It is divisible by 4 and not divisible by 100, **or**
     - It is divisible by 400.

3. If the date is valid, display: `"The date <day>/<month>/<year> is valid."`
4. If the date is invalid, display: `"The date <day>/<month>/<year> is not valid."`

Example:

- **Input:**

  ```
  Enter day: 29
  Enter month: 2
  Enter year: 2024
  ```

  Output:

  ```
  The date 29/2/2024 is valid.
  ```

- **Input:**

  ```
  Enter day: 30
  Enter month: 2
  Enter year: 2021
  ```

  Output:

  ```
  The date 30/2/2021 is not valid.
  ```

- **Input:**

  ```
  Enter day: 31
  Enter month: 11
  Enter year: 2023
  ```

  Output:

  ```
  The date 31/11/2023 is not valid.
  ```

Hints:

- Validate the month first, ensuring it falls within 1–12.
- Use conditional statements to check the number of days for each month.
- For February, consider the rules for leap years.
- Handle edge cases, such as invalid day/month/year input.

12. Voting Eligibility Checker

Write a program in the C programming language to determine whether a person is eligible to vote based on their age.

Detailed Requirements:

1. The user will input their age (an integer).
2. The program must determine:
   - If the age is 18 or above, display: `"You are eligible to vote."`
   - If the age is less than 18 but positive, display: `"You are not eligible to vote. You need to wait <years> more years."`
   - If the age is invalid (e.g., negative or zero), display: `"Invalid age entered. Please enter a positive number."`

Example:

- **Input:**

  `Enter your age: 25`

  Output:

  `You are eligible to vote.`

- **Input:**

  `Enter your age: 15`

  Output:

  `You are not eligible to vote. You need to wait 3 more years.`

- **Input:**

  `Enter your age: -5`

  Output:

  `Invalid age entered. Please enter a positive number.`

- **Input:**

  `Enter your age: 0`

  Output:

  `Invalid age entered. Please enter a positive number.`

Hints:

- Use conditional statements (`if-else`) to check the age conditions.
- Handle edge cases like invalid input (negative or zero).
- Calculate the number of years left for voting eligibility in cases where the age is below 18.

13. Find the maximum between two numbers

Write a program in the C programming language to find the maximum of two numbers entered by the user.

Detailed Requirements:

1. The user will input two numbers (integers or floating-point values).
2. The program must determine:
   - If the first number is greater, display: `"The maximum number is: <first number>."`
   - If the second number is greater, display: `"The maximum number is: <second number>."`
   - If both numbers are equal, display: `"Both numbers are equal."`

Example:

- **Input:**

  `Enter the first number: 20`

  `Enter the second number: 15`

  Output:

  `The maximum number is: 20.`

- **Input:**

  `Enter the first number: -5`

  `Enter the second number: -10`

  Output:

  `The maximum number is: -5.`

- **Input:**

  `Enter the first number: 7`

  `Enter the second number: 7`

  Output:

  `Both numbers are equal.`

Hints:

- Use conditional statements (`if-else`) or the ternary operator (`?:`) to compare the two numbers.

- Handle equality explicitly for clarity in output.
- Provide clear prompts and outputs for user interaction.

14. Find the maximum between the three numbers

Write a program in the C programming language to find the maximum of three numbers entered by the user.

Detailed Requirements:

1. The user will input three numbers (integers or floating-point values).
2. The program must determine and display the maximum number among the three.
3. If all three numbers are equal, display: `"All three numbers are equal."`

Example:

- **Input:**

  ```
  Enter the first number: 10
  Enter the second number: 20
  Enter the third number: 15
  ```

  Output:

  ```
  The maximum number is: 20.
  ```

- **Input:**

  ```
  Enter the first number: -5
  Enter the second number: -10
  Enter the third number: -3
  ```

  Output:

  ```
  The maximum number is: -3.
  ```

- **Input:**

  ```
  Enter the first number: 7
  Enter the second number: 7
  Enter the third number: 7
  ```

  Output:

  ```
  All three numbers are equal.
  ```

Hints:

- Use nested conditional statements (`if-else`) or logical operators to compare the three numbers.
- Handle the case where all numbers are equal explicitly for clear output.
- Provide clear prompts for the user to input the numbers and display results

15. Check whether a number is negative, positive or zero

Write a program in the C programming language to determine whether a given number is negative, positive, or zero.

Detailed Requirements:

1. The user will input a number (integer or floating-point).
2. The program must determine and display:
   - If the number is greater than zero, display: `"The number is positive."`
   - If the number is less than zero, display: `"The number is negative."`
   - If the number is zero, display: `"The number is zero."`

Example:

- **Input:**

  `Enter a number: 25`

  Output:

  `The number is positive.`
- **Input:**

  `Enter a number: -10`

  Output:

  `The number is negative.`
- **Input:**

  `Enter a number: 0`

  Output:

  `The number is zero.`

Hints:

- Use conditional statements (`if-else`) to evaluate the number.
- Include edge cases like zero to ensure correct results.
- Provide clear prompts for input and detailed outputs.

16. Check whether a number is divisible by 5 and 11 or not

Write a program in the C programming language to check whether a given number is divisible by both 5 and 11.

Detailed Requirements:

1. The user will input a number (integer).
2. The program must determine and display:
   - If the number is divisible by both 5 and 11, display: `"The number is divisible by both 5 and 11."`
   - If the number is not divisible by both, display: `"The number is not divisible by both 5 and 11."`

Example:

- **Input:**

  `Enter a number: 55`

  Output:

  `The number is divisible by both 5 and 11.`

- **Input:**

  `Enter a number: 20`

  Output:

  `The number is not divisible by both 5 and 11.`

- **Input:**

  `Enter a number: 121`

  Output:

  `The number is not divisible by both 5 and 11.`

Hints:

- Use the modulus operator (`%`) to check for divisibility:
  - A number is divisible by 5 if `number % 5 == 0`.
  - A number is divisible by 11 if `number % 11 == 0`.
- Use a logical `AND` operator (`&&`) to check both conditions simultaneously.
- Provide clear prompts for input and detailed outputs.

17. Find whether a number is even or odd

Write a program in the C programming language to determine whether a given number is even or odd.

Detailed Requirements:

1. The user will input a number (integer).
2. The program must determine and display:
   - If the number is divisible by 2, display: `"The number is even."`
   - Otherwise, display: `"The number is odd."`

Example:

- **Input:**

  `Enter a number: 12`

  Output:

  `The number is even.`
- **Input:**

  `Enter a number: 7`

  Output:

  `The number is odd.`
- **Input:**

  `Enter a number: 0`

  Output:

  `The number is even.`

Hints:

- Use the modulus operator (`%`) to check for divisibility:

    - A number is even if `number % 2 == 0`.
    - A number is odd otherwise.
- Provide clear prompts for the user to enter the number and detailed outputs for clarity

18. Check whether a year is a leap year or not.

Write a program in the C programming language to determine whether a given year is a leap year or not.

Detailed Requirements:

1. The user will input a year (integer).
2. The program must determine and display:
    - A year is a leap year if:
        - It is divisible by 4 **and** not divisible by 100, **or**
        - It is divisible by 400.
    - If the year is a leap year, display: `"The year <year> is a leap year."`
    - Otherwise, display: `"The year <year> is not a leap year."`

Example:

- **Input:**

  `Enter a year: 2024`

  Output:

  `The year 2024 is a leap year.`
- **Input:**

  `Enter a year: 1900`

  Output:

  `The year 1900 is not a leap year.`
- **Input:**

  `Enter a year: 2000`

  Output:

`The year 2000 is a leap year.`

Hints:

- Use the modulus operator (`%`) to check divisibility:
  - `year % 4 == 0` checks if the year is divisible by 4.
  - `year % 100 != 0` ensures the year is not divisible by 100 (for non-century years).
  - `year % 400 == 0` handles the exception for years divisible by 400.
- Use conditional statements (`if-else`) to evaluate the conditions.
- Provide clear prompts for input and detailed outputs.

19. Check whether a character is an alphabet or not

Write a program in the C programming language to check whether a given character is an alphabet or not.

Detailed Requirements:

1. The user will input a single character.
2. The program must determine and display:
   - If the character is an alphabet (either uppercase `A-Z` or lowercase `a-z`), display: `"The character '<character>' is an alphabet."`
   - Otherwise, display: `"The character '<character>' is not an alphabet."`

Example:

- **Input:**

  `Enter a character: A`

  Output:

  `The character 'A' is an alphabet.`

- **Input:**

  `Enter a character: z`

  Output:

  `The character 'z' is an alphabet.`

- **Input:**

  `Enter a character: 5`

  Output:

  `The character '5' is not an alphabet.`

- **Input:**

  `Enter a character: @`

  Output:

  `The character '@' is not an alphabet.`

Hints:

- Use the ASCII values to check the range:
  - `A-Z` corresponds to ASCII values 65–90.
  - `a-z` corresponds to ASCII values 97–122.
- Use conditional statements (`if-else`) to evaluate the input.
- Provide clear prompts for input and detailed outputs for clarity.

20. Input any alphabet and check whether it is vowel or consonant

Write a program in the C programming language to determine whether a given alphabet is a vowel or a consonant.

Detailed Requirements:

1. The user will input a single alphabet character.
2. The program must determine and display:
   - If the character is a vowel (`a, e, i, o, u` or `A, E, I, O, U`), display: `"The character '<character>' is a vowel."`
   - If the character is a consonant, display: `"The character '<character>' is a consonant."`
   - If the input is not an alphabet, display: `"Invalid input. Please enter an alphabet."`

Example:

- **Input:**

`Enter a character: A`

Output:

`The character 'A' is a vowel.`

- **Input:**

  `Enter a character: x`

  Output:

  `The character 'x' is a consonant.`

- **Input:**

  `Enter a character: 7`

  Output:

  `Invalid input. Please enter an alphabet.`

Hints:

- Use the ASCII values to check if the input is an alphabet:
  - `A-Z` corresponds to ASCII values 65–90.
  - `a-z` corresponds to ASCII values 97–122.
- Use conditional statements (`if-else` or `switch-case`) to check for vowels:
  - Vowels are `a, e, i, o, u` (both uppercase and lowercase).
- Provide clear prompts for input and detailed outputs for clarity.

21. Input any character and check whether it is the alphabet, digit or special character

Write a program in the C programming language to determine whether a given character is an alphabet, a digit, or a special character.

Detailed Requirements:

1. The user will input a single character.
2. The program must determine and display:

- If the character is an alphabet (either uppercase `A-Z` or lowercase `a-z`), display: `"The character '<character>' is an alphabet."`
- If the character is a digit (`0-9`), display: `"The character '<character>' is a digit."`
- If the character is neither, display: `"The character '<character>' is a special character."`

Example:

- **Input:**

  `Enter a character: A`

  Output:

  `The character 'A' is an alphabet.`

- **Input:**

  `Enter a character: 7`

  Output:

  `The character '7' is a digit.`

- **Input:**

  `Enter a character: @`

  Output:

  `The character '@' is a special character.`

Hints:

- Use the ASCII values to categorize the character:
  - Alphabets: `A-Z` (65–90) or `a-z` (97–122).
  - Digits: `0-9` (48–57).
  - Special characters: Anything not in the above ranges.
- Use conditional statements (`if-else`) for comparisons.
- Provide clear prompts and outputs for user interaction

22. Check whether a character is an uppercase or lowercase alphabet

Write a program in the C programming language to determine whether a given alphabet character is uppercase or lowercase.

Detailed Requirements:

1. The user will input a single character.
2. The program must determine and display:
   - If the character is an uppercase alphabet (`A-Z`), display: `"The character '<character>' is an uppercase alphabet."`
   - If the character is a lowercase alphabet (`a-z`), display: `"The character '<character>' is a lowercase alphabet."`
   - If the input is not an alphabet, display: `"The character '<character>' is not an alphabet."`

Example:

- **Input:**

  `Enter a character: A`

  Output:

  `The character 'A' is an uppercase alphabet.`

- **Input:**

  `Enter a character: z`

  Output:

  `The character 'z' is a lowercase alphabet.`

- **Input:**

  `Enter a character: 5`

  Output:

  `The character '5' is not an alphabet.`

Hints:

- Use the ASCII values to check:
  - Uppercase letters: `A-Z` (ASCII 65–90).
  - Lowercase letters: `a-z` (ASCII 97–122).
- Use conditional statements (`if-else`) for classification.
- Ensure to handle invalid inputs, like digits or special characters, with appropriate output messages.

23. Input week number and print weekday.

Write a program in the C programming language to display the corresponding weekday for a given week number.

Detailed Requirements:

1. The user will input a week number (integer between 1 and 7).
2. The program must determine and display the corresponding weekday:
   - `1` for "Monday"
   - `2` for "Tuesday"
   - `3` for "Wednesday"
   - `4` for "Thursday"
   - `5` for "Friday"
   - `6` for "Saturday"
   - `7` for "Sunday"
3. If the input is not between 1 and 7, display: `"Invalid input. Please enter a number between 1 and 7."`

Example:

- **Input:**

  `Enter a week number: 3`

  Output:

  `The weekday is Wednesday.`

- **Input:**

  `Enter a week number: 7`

  Output:

  `The weekday is Sunday.`

- **Input:**

  `Enter a week number: 8`

  Output:

`Invalid input. Please enter a number between 1 and 7.`

Hints:

- Use a `switch-case` statement to map week numbers to weekdays.
- Ensure proper validation to handle invalid inputs outside the range 1–7.
- Provide clear prompts and output messages for user interaction.

24. Input month number and print number of days in that month.

Write a program in the C programming language to display the number of days in a given month based on its month number.

Detailed Requirements:

1. The user will input a month number (integer between 1 and 12).
2. The program must determine and display:
   - `31 days` for months January (1), March (3), May (5), July (7), August (8), October (10), December (12).
   - `30 days` for months April (4), June (6), September (9), November (11).
   - `28 days` for February (2) in common years and `29 days` in leap years.
3. If the input is not between 1 and 12, display: `"Invalid input. Please enter a number between 1 and 12."`
4. For February, determine whether the year is a leap year using these rules:
   - A year is a leap year if:
     - It is divisible by 4 and not divisible by 100, **or**
     - It is divisible by 400.

Example:

- **Input:**

  `Enter a month number: 2`

  `Enter a year: 2024`

  Output:

  `The month has 29 days.`

- **Input:**

  `Enter a month number: 6`

  Output:

  `The month has 30 days.`

- **Input:**

  `Enter a month number: 13`

  Output:

  `Invalid input. Please enter a number between 1 and 12.`

Hints:

- Use a `switch-case` statement for month-based decision-making.
- For February, include an additional check for leap years.
- Ensure proper input validation to handle invalid month numbers.

25. Count the total number of notes in a given amount

Write a program in the C programming language to calculate the minimum number of currency notes required for a given amount. The program should consider the denominations: 2000, 500, 200, 100, 50, 20, 10, 5, 2, and 1.

Detailed Requirements:

1. The user will input an amount (a positive integer).
2. The program must calculate and display the minimum number of notes required for the given amount, along with the count of each denomination.
3. The denominations are: 2000, 500, 200, 100, 50, 20, 10, 5, 2, and 1.

Example:

- **Input:**

  `Enter the amount: 2783`

  Output:

```
2000: 1
```

```
500: 1
200: 1
50: 1
20: 1
10: 1
2: 1
1: 1
Total notes: 8
```

- **Input:**

  `Enter the amount: 350`

  Output:

```
200: 1
100: 1
50: 1
Total notes: 3
```

Hints:

- Use a greedy algorithm:
  - Start with the highest denomination and use as many as possible.
  - Move to the next smaller denomination until the amount becomes zero.
- Use a loop or sequential conditions to iterate through the denominations.
- Keep track of the total count of notes.

26. Input angles of a triangle and check whether the triangle is valid or not.

Write a program in the C programming language to determine whether a triangle is valid based on its three angles.

Detailed Requirements:

1. The user will input three angles of a triangle (integers or floating-point values).
2. The program must check the validity of the triangle using the following conditions:
   - The sum of the three angles must be exactly 180 degrees.
   - Each angle must be greater than 0.
3. If the triangle is valid, display: `"The triangle is valid."`
4. If the triangle is not valid, display: `"The triangle is not valid."`

Example:

- **Input:**

  ```
  Enter the first angle: 60
  Enter the second angle: 60
  Enter the third angle: 60
  ```

  Output:

  ```
  The triangle is valid.
  ```

- **Input:**

  ```
  Enter the first angle: 90
  Enter the second angle: 45
  Enter the third angle: 50
  ```

  Output:

  ```
  The triangle is not valid.
  ```

- **Input:**

  ```
  Enter the first angle: 0
  Enter the second angle: 90
  Enter the third angle: 90
  ```

  Output:

  ```
  The triangle is not valid.
  ```

Hints:

- Use conditional statements (`if-else`) to check the validity.
- Ensure the input values for angles are positive.
- Check the sum of the angles using `angle1 + angle2 + angle3 == 180`.

27. Input all sides of a triangle and check whether the triangle is valid or not.

Write a program in the C programming language to determine whether a triangle is valid based on the lengths of its three sides.

Detailed Requirements:

1. The user will input the three sides of a triangle (positive integers or floating-point numbers).
2. The program must check the validity of the triangle using the triangle inequality theorem:
   - The sum of any two sides must be greater than the third side.
   - This must be true for all three combinations of sides:
     - `side1 + side2 > side3`
     - `side1 + side3 > side2`
     - `side2 + side3 > side1`
3. If the triangle is valid, display: `"The triangle is valid."`
4. If the triangle is not valid, display: `"The triangle is not valid."`

Example:

- **Input:**

  ```
  Enter the first side: 3
  Enter the second side: 4
  Enter the third side: 5
  ```

  Output:

  ```
  The triangle is valid.
  ```

- **Input:**

  ```
  Enter the first side: 1
  Enter the second side: 2
  Enter the third side: 3
  ```

  Output:

  ```
  The triangle is not valid.
  ```

- **Input:**

  `Enter the first side: 7`

  `Enter the second side: 10`

  `Enter the third side: 5`

  Output:

  `The triangle is valid.`

Hints:

- Use conditional statements (`if-else`) to check the validity.
- Ensure the input values for sides are positive.
- Apply the triangle inequality theorem:
  - Check all three conditions mentioned above.
- Provide clear prompts for the user to input the side lengths and display the results clearly.

28. Check whether the triangle is an equilateral, isosceles or scalene triangle.

Write a program in the C programming language to determine whether a triangle is equilateral, isosceles, or scalene based on the lengths of its sides.

Detailed Requirements:

1. The user will input the three sides of a triangle (positive integers or floating-point numbers).
2. The program must first validate the triangle using the triangle inequality theorem:
   - The sum of any two sides must be greater than the third side.
   - If the triangle is not valid, display: `"The triangle is not valid."`
3. If the triangle is valid, determine and display its type:
   - **Equilateral Triangle:** All three sides are equal. Display: `"The triangle is equilateral."`
   - **Isosceles Triangle:** Two sides are equal. Display: `"The triangle is isosceles."`

- **Scalene Triangle:** All three sides are different. Display: `"The triangle is scalene."`

Example:

- **Input:**

  `Enter the first side: 5`

  `Enter the second side: 5`

  `Enter the third side: 5`

  Output:

  `The triangle is equilateral.`

- **Input:**

  `Enter the first side: 7`

  `Enter the second side: 7`

  `Enter the third side: 5`

  Output:

  `The triangle is isosceles.`

- **Input:**

  `Enter the first side: 3`

  `Enter the second side: 4`

  `Enter the third side: 5`

  Output:

  `The triangle is scalene.`

- **Input:**

  `Enter the first side: 1`

  `Enter the second side: 2`

  `Enter the third side: 3`

  Output:

  `The triangle is not valid.`

Hints:

1. Validate the triangle first using the triangle inequality theorem:
   - `side1 + side2 > side3`
   - `side1 + side3 > side2`
   - `side2 + side3 > side1`
2. If the triangle is valid:

- Check for **Equilateral Triangle**: `side1 == side2 && side2 == side3`
- Check for **Isosceles Triangle**: `side1 == side2 || side2 == side3 || side1 == side3`
- Otherwise, it is a **Scalene Triangle**.

3. Use nested `if-else` statements for validation and classification.

29. Find all roots of a quadratic equation

Write a program in the C programming language to find all roots of a quadratic equation of the form:

$$ax2 + bx + c = 0ax^2 + bx + c = 0$$

$$ax2 + bx + c = 0$$

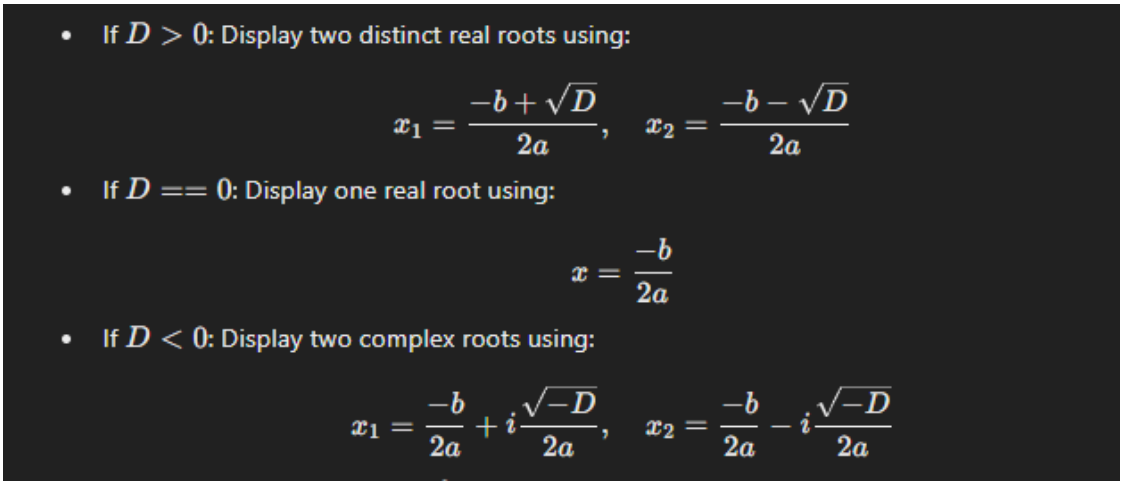
where `a, b, c` are coefficients provided by the user.

Detailed Requirements:

1. The user will input the coefficients a, b, and c.
2. The program must compute the discriminant (D) using the formula:

$$D = b2 - 4acD = b^2 - 4ac$$

1. Based on the value of D:



2. Handle the edge case where a=0 (not a quadratic equation), and display an appropriate message.

Example:

- **Input:**

  ```
  Enter coefficients a, b, c: 1, -3, 2
  ```

  Output:

  ```
  The roots are 2 and 1.
  ```

- **Input:**

  ```
  Enter coefficients a, b, c: 1, -2, 1
  ```

  Output:

  ```
  The root is 1.
  ```

- **Input:**

  ```
  Enter coefficients a, b, c: 1, 1, 1
  ```

  Output:

  ```
  The roots are -0.5 + 0.866i and -0.5 - 0.866i.
  ```

- **Input:**

  ```
  Enter coefficients a, b, c: 0, 2, 1
  ```

  Output:

  ```
  This is not a quadratic equation.
  ```

Hints:

1. Use the discriminant formula $D = b^2 - 4ac$ to classify the roots.
2. Use the `sqrt` function from `<math.h>` to compute the square root.
3. For $D < 0$, calculate and display the real and imaginary parts separately.
4. Validate user input to ensure $a \neq 0$.

30. Calculate profit or loss.

Write a program in the C programming language to calculate whether a transaction resulted in a profit or a loss, and determine its amount.

Detailed Requirements:

1. The user will input two values:
   - Cost price (CP): The amount spent to acquire the item.
   - Selling price (SP): The amount at which the item was sold.
2. The program must determine:
   - If $SP > CP$, calculate profit using the formula:

     $$\text{Profit} = SP - CP$$

     and display: `"You made a profit of <amount>."`
   - If $SP < CP$, calculate loss using the formula:

     $$\text{Loss} = CP - SP$$

     and display: `"You incurred a loss of <amount>."`
   - If $SP == CP$, display: `"There is no profit or loss."`

Example:

- **Input:**

  ```
  Enter cost price: 500
  Enter selling price: 700
  ```

  Output:

  ```
  You made a profit of 200.
  ```

- **Input:**

  ```
  Enter cost price: 800
  Enter selling price: 500
  ```

  Output:

  ```
  You incurred a loss of 300.
  ```

- **Input:**

  ```
  Enter cost price: 1000
  Enter selling price: 1000
  ```

Output:

```
There is no profit or loss.
```

Hints:

- Use conditional statements (`if-else`) to compare SP and CP.
- Ensure that the program handles edge cases, such as CP or SP being zero.
- Provide clear prompts for user input and detailed output messages.