Day 6	Type Casting and Conditional Statement

Objectives	Understand the limitations and errors associated with type casting between incompatible data types and develop programs using conditional statements to control program flow for decision making.
Outcomes	Students will be able to 1.Able to write programs that make decisions based on different conditions. 2. Improve the skills in handling different cases and edge conditions. 3.Write efficient conditional logic to optimize performance.

1.Number Sign Checker

Create an utility for financial analysts who need to quickly determine the nature of a number's value (positive, negative, or zero) in their calculations. Check if a number is positive, negative, or zero.

Constraints:

The input should be an integer.

The number should be within the range of typical 32-bit integers (-2,147,483,648 to 2,147,483,647).

Test Case 1:
Input: Enter an integer: -987654321
Output: The number is negative

Test Case 2:
Input: Enter an integer: 123456789
Output: The number is positive

Test Case 3:
Input: Enter an integer: 0
Output: The number is zero

Test Case 4:
Input: Enter an integer: -1
Output: The number is negative

Test Case 5:
Input: Enter an integer: 1
Output: The number is positive

Test Case 6:
Input: Enter an integer: -2147483648
Output: The number is negative

Test Case 7:
Input: Enter an integer: 2147483647
Output: The number is positive

Test Case 8:
Input: Enter an integer: -123
Output: The number is negative

Test Case 9:
Input: Enter an integer: -45
Output: The number is negative

Test Case 10:
Input: Enter an integer: 23
Output: The number is positive

2.Day Name Checker

Develop an event scheduling application for a business. The application needs to differentiate between weekdays and weekends because certain activities, meetings, and maintenance tasks should only be scheduled on weekdays, while social events and leisure activities are preferred for weekends.

Develop a C program that differentiates between weekdays and weekends based on user input, while considering case sensitivity for day names.

Constraints:

Case Sensitivity:

Exact case-sensitive comparisons of input day names

Input Validation:

Valid inputs include recognized day names ("Monday" to "Sunday" and their abbreviations).

Invalid inputs should prompt an appropriate error message ("Invalid day name").

Edge Cases Handling:

Handle mixed-case inputs correctly (e.g., "MoNdAy", "SaTuRdAy").

Validate both full day names ("Monday", "Saturday") and their abbreviations ("Mon", "Sat").

TEST CASE 1:
Input: Enter a day name: Monday
Output:
Monday is a weekday.

TEST CASE 2:
Input: Enter a day name: monday
Output:
Invalid day name

TEST CASE 3:
Input: Enter a day name: Wed
Output:
Wed is a weekday.

TEST CASE 4:
Input: Enter a day name: Saturday
Output:
Saturday is a weekend day.

TEST CASE 5:
Input: Enter a day name: sunday
Output:
Invalid day name

TEST CASE 6:
Input: Enter a day name: Tue
Output:
Tue is a weekday.

TEST CASE 7:
Input: Enter a day name: Wednesday
Output:
Wednesday is a weekday.

TEST CASE 8:
Input: Enter a day name: Sat

Output:
Sat is a weekend day.

TEST CASE 9:
Input:
Enter a day name: tUeSdAy
Output:
Invalid day name

TEST CASE 10:
Input: 123abc
Output:
Invalid day name

3.Valid Triangle Checker

You are tasked with developing a program to assist in architectural design. Write a C program that verifies whether three given lengths can form a valid triangle based on the triangle inequality theorem. Ensure the program accurately determines if the sum of any two sides is greater than the length of the remaining side, providing clear feedback on validity.

Constraints:

Side Length Constraints:
Type: Integer
Range: 1 to 2,147,483,647 (standard int range in C)
Description: Side lengths should be positive integers, as negative or zero values do not make sense in the context of triangle sides.

TEST CASE 1:
Input:
Enter the lengths of the three sides of the triangle: 5 6 9
Output:
Triangle with sides 5, 6, 9 is Valid Triangle.

TEST CASE 2:
Input: Enter the lengths of the three sides of the triangle: 12 2 5
Output:
Triangle with sides 12, 2, 5 is Invalid Triangle.

TEST CASE 3:

Input: Enter the lengths of the three sides of the triangle: 45 52 87
Output:
Triangle with sides 45, 52, 87 is Valid Triangle.

TEST CASE 4:
Input: Enter the lengths of the three sides of the triangle: 35 22 96
Output:
Triangle with sides 35, 22, 96 is Invalid Triangle.

TEST CASE 5:
Input: Enter the lengths of the three sides of the triangle: 4 4 4
Output:
Triangle with sides 4, 4, 4 is Valid Triangle.

TEST CASE 6:
Input:Enter the lengths of the three sides of the triangle: -5 -5 -5
Output:
Error. Length of sides of triangle cannot be negative or zero.

TEST CASE 7:
Input: Enter the lengths of the three sides of the triangle: 100 45 52
Output:
Triangle with sides 100, 45, 52 is Invalid Triangle.

TEST CASE 8:
Input: Enter the lengths of the three sides of the triangle: 0 3 4
Output:
Error. Length of sides of triangle cannot be negative or zero.

TEST CASE 9:
Input:
Enter the lengths of the three sides of the triangle: 5 5 5
Output:
Triangle with sides 5, 5, 5 is Valid Triangle.

TEST CASE 10:
Input: Enter the lengths of the three sides of the triangle: 0 0 0
Output:
Error. Length of sides of triangle cannot be negative or zero.

4.Age Group Categorizer

To assist a healthcare provider in categorizing patients into specific age groups for personalized medical care plans, develop a robust C program that prompts the user to input their age, validates the input to ensure it falls within acceptable ranges, and categorizes the age into one of the specified groups. The program should handle edge cases such as negative ages or non-integer inputs gracefully, providing informative messages for each category.

Constraints:

Child: Individuals aged from 0 to 12 years.
Teenager: Individuals aged from 13 to 19 years.
Adult: Individuals aged from 20 to 59 years.
Senior: Individuals aged 60 years and above.
In valid age category negative values

TEST CASE 1:
Input:
Enter your age: 15.0
Output:
Based on your age of 15 years, you belong to the Teenager category.

TEST CASE 2:
Input: Enter your age: 61.0
Output:
Based on your age of 61 years, you belong to the Senior category.

TEST CASE 3:
Input: Enter your age: -10.0
Output:
Invalid age category.

TEST CASE 4:
Input: Enter your age: 0.0
Output:
Based on your age of 0 years, you belong to the Child category.

TEST CASE 5:
Input: Enter your age: 52.0

Output:
Based on your age of 52 years, you belong to the Adult category.
TEST CASE 6:
Input: Enter your age: 45.0
Output:
Based on your age of 45 years, you belong to the Adult category.

TEST CASE 7:
Input: Enter your age: 10.0
Output:
Based on your age of 10 years, you belong to the Child category.

TEST CASE 8:
Input: Enter your age: 18.5
Output:
Invalid age category.

TEST CASE 9:
Input:
Enter your age: 0.5
Output:
Invalid age category.

TEST CASE 10:
Input:
Enter your age: fifteen
Output:
Invalid input. Please enter a valid age.

5.Traffic Light Simulator

Write a C program that simulates a traffic light by taking the current light color as input and printing the appropriate action based on the following rules:
If the current light is "Red", print "Stop".
If the current light is "Yellow", print "Proceed with caution".
If the current light is "Green", print "Go".

Constraints

- **Input Size**: The program reads up to 19 characters for the light color to avoid buffer overflow.

- **Case Sensitivity**: The program expects exact matches ("Red", "Yellow", "Green"). Inputs with different cases or extra characters are considered invalid.

Test Case 1:
Input: Red
Output: Stop

Test Case 2:
Input: Green
Output: Go

Test Case 3:
Input: Yellow
Output: Proceed with caution

Test Case 4:
Input: Blue
Output: Invalid input

Test Case 5:
Input: red
Output: Invalid input

Test Case 6:
Input: GrEEn
Output: Invalid input

Test Case 7:
Input: Red
Output: Invalid input

Test Case 8:
Input: GrEEn
Output: Invalid input

Test Case 9:
Input: Orange
Output: Invalid input

Test Case 10:
Input: Red!
Output: Invalid input

6.Simple Discount Calculator

A shopping application allows users to calculate discounts or pricing adjustments based on their inputs. The application should prompt users to enter specific values and operations to compute final prices or discounts.

Constraints

The program should validate user inputs for original_price and discount_percentage to ensure they are valid float values and within reasonable ranges (e.g., non-negative values for original_price and non-negative up to 100% or zero for discount_percentage).

Test Case 1:
Input:
Original Price = $100, Discount Percentage (0 for no discount): = 20
Output:
Original Price: $100.00
Discount Percentage: 20.00%
Discounted Price: $80.00

Test Case 2:
Input:
Original Price = $50, Discount Percentage (0 for no discount): = 0
Output:
Original Price: $50.00
No Discount Applied
Discounted Price: $50.00

Test Case 3:
Input:
Original Price = $999.99, Discount Percentage (0 for no discount): = 15
Output:
Original Price: $999.99
Discount Percentage: 15.00%
Discounted Price: $849.99

Test Case 4:
Input:

original price: $-50
Output:
Invalid input. Please enter a valid non-negative price.

Test Case 5:
Input:
original price: $250
discount percentage (0 for no discount): 100
Output:
Original Price: $250.00
Discount Percentage: 100.00%
Discounted Price: $0.00

Test Case 6:
Input:
Original Price = $80, Discount Percentage (0 for no discount): = -10
Output:
Invalid input. Please enter a valid non-negative discount percentage.

Test Case 7:
Input:
original price: $5000
Output:
discount percentage (0 for no discount): 1000
Invalid input. Please enter a valid discount percentage.

Test Case 8:
Input:
Original Price = $120, Discount Percentage (0 for no discount): = 12.5
Output:
Original Price: $120.00
Discount Percentage: 12.50%
Discounted Price: $105.00

Test Case 9:
Input:
Original Price = $0, Discount Percentage (0 for no discount): = 15
Output:
Original Price: $0.00

Discount Percentage: 15.00%
Discounted Price: $0.00

Test Case 10:
Input:
Original Price = $999999.99, Discount Percentage (0 for no discount): = 10
Output:
Original Price: $1000000.00
Discount Percentage: 10.00%
Discounted Price: $900000.00

7.Season Identifier

Develop a travel planning application that provides seasonal recommendations for various destinations. Users input the month they plan to travel, and the application suggests activities, clothing, and places to visit based on the season. Write a C program that takes a month number as input and identifies the corresponding season. This helps in tailoring the travel itinerary and providing useful suggestions to the user.

Constraints:

Seasonal Divisions:

- Spring: March (3), April (4), May (5)
- Summer: June (6), July (7), August (8)
- Fall: September (9), October (10), November (11)
- Winter: December (12), January (1), February (2)

Test Case 1
Input: Enter the month number (1-12): 3
Output:
You are traveling in Spring.
Activities: Enjoy blooming flowers, hiking.
Clothing: Light jackets, layers.
Places to visit: Gardens, national parks.

Test Case 2
Input: Enter the month number (1-12): 4
Output: You are traveling in Spring.
Activities: Enjoy blooming flowers, hiking.
Clothing: Light jackets, layers.
Places to visit: Gardens, national parks.

Test Case 3
Input: Enter the month number (1-12): 6
Output:
You are traveling in Summer.
Activities: Beach activities, water sports.
Clothing: Swimsuits, light clothing.
Places to visit: Beach resorts, lakeside destinations.

Test Case 4
Input: Enter the month number (1-12): 12
Output:
You are traveling in Winter.
Activities: Skiing, snowboarding.
Clothing: Winter coats, gloves, hats.
Places to visit: Ski resorts, winter wonderlands.

Test Case 5
Input: Enter the month number (1-12): 0
Output: Invalid month input. Please enter a month number between 1 and 12.

Test Case 6
Input: Enter the month number (1-12): 1
Output:
You are traveling in Winter.
Activities: Skiing, snowboarding.
Clothing: Winter coats, gloves, hats.
Places to visit: Ski resorts, winter wonderlands.

Test Case 7
Input: Enter the month number (1-12): -2
Output:
Invalid month input. Please enter a month number between 1 and 12.

Test Case 8
Input: Enter the month number (1-12): 5
Output:
You are traveling in Spring.
Activities: Enjoy blooming flowers, hiking.

Clothing: Light jackets, layers.
Places to visit: Gardens, national parks.

Test Case 9
Input: Enter the month number (1-12): 10
Output:
You are traveling in Fall.
Activities: Apple picking, leaf peeping.
Clothing: Sweaters, jeans.
Places to visit: Orchards, countryside.

Test Case 10
Input: Enter the month number (1-12): 11
Output:
You are traveling in Fall.
Activities: Apple picking, leaf peeping.
Clothing: Sweaters, jeans.
Places to visit: Orchards, countryside.

8.Budgeting Tool Development

Develop a budgeting tool in C that categorizes expenses into different spending ranges based on the transaction amount. The program should prompt users to input the transaction amount and then categorize it into one of three predefined spending ranges:

1. **0-50**: Low expense category.
2. **51-100**: Medium expense category.
3. **Above 100**: High expense category.

Constraints

The program validates that the transaction amount entered is a non-negative numeric value.
It categorizes the expense into one of the predefined spending ranges and provides clear output formatted to two decimal places.

Test Case 1
Input:0
Output
Transaction Amount: $0.00
Category: Low expense category

Test Case 2
Input:100
Output
Transaction Amount: $100.00
Category: Medium expense category

Test Case 3
Input:1000
Output
Transaction Amount: $1000.00
Category: High expense category

Test Case 4
Input:1000.5
Output
Transaction Amount: $1000.50
Category: High expense category

Test Case 5
Input:-2000
Output
Invalid input. Please enter a non-negative numeric value.

Test Case 6
Input:100.20
Output
Transaction Amount: $100.20
Category: High expense category

Test Case 7
Input:52
Output
Transaction Amount: $52.00
Category: Medium expense category

Test Case 8
Input:15
Output
Transaction Amount: $15.00
Category: Low expense category

Test Case 9
Input:-5
Output
Invalid input. Please enter a non-negative numeric value.

Test Case 10
Input:100000.25689
Output
Transaction Amount: $100000.26
Category: High expense category

9.Temperature State

Develop a scientific tool in C that analyzes the state of water based on temperature readings in Celsius. The program should classify temperatures into three distinct states of water:

1. **Solid State**: Water freezes below 0 degrees Celsius.
2. **Liquid State**: Water exists in its liquid form between 0 and 100 degrees Celsius.
3. **Gas State**: Water evaporates above 100 degrees Celsius.

Constraints

- Temperature is negative degrees Celsius (solid state).
- Temperature just below 100 degrees Celsius (liquid state).
- Temperature above 100 degrees Celsius (gas state).

Ensure the program correctly categorizes these edge values without error.

Test Case 1
Input: Enter the temperature in Celsius: -12
Output: Water is in solid state.

Test Case 2
Input: Enter the temperature in Celsius:0
Output: Water is in solid state.

Test Case 3
Input: Enter the temperature in Celsius:15
Output: Water is in liquid state.

Test Case 4
Input: Enter the temperature in Celsius:100
Output: Water is in liquid state.

Test Case 5
Input: Enter the temperature in Celsius:99.9
Output: Water is in liquid state.

Test Case 6
Input: Enter the temperature in Celsius:0.12
Output: Water is in liquid state.

Test Case 7
Input: Enter the temperature in Celsius:Eighty three
Output: Invalid input. Please enter a valid numeric temperature.

Test Case 8
Input: Enter the temperature in Celsius:155.2354
Output: Water is in gas state.

Test Case 9
Input: Enter the temperature in Celsius:156
Output: Water is in a gas state.

Test Case 10
Input: Enter the temperature in Celsius:1.235689
Output: Water is in liquid state.

10.Quadrant Locator

Develop a program to assist in plotting points on a Cartesian plane. The program should take coordinates (x, y) as input and determine which quadrant the point lies in. Prompt the user to enter the coordinates (x, y).Determines which quadrant the point (x, y) falls into using if-else statements.Print the quadrant number or a message indicating if the point lies on an axis or at the origin.

Constraints:

Quadrant I: Both x and y are positive numbers
Quadrant II: where x is negative and y is positive
Quadrant III: Both x and y are negative numbers

Quadrant IV: where x is positive and y is negative

Test Case 1
Input:
Enter the coordinates (x, y): 2 3
Output
The point (2, 3) lies in Quadrant I.

Test Case 2
Input:
Enter the coordinates (x, y): 18 19
Output:
The point (18, 19) lies in Quadrant I.

Test Case 3
Input:
Enter the coordinates (x, y): -2 5
Output:
The point (-2, 5) lies in Quadrant II.

Test Case 4
Input:
Enter the coordinates (x, y): -5 -5
Output:
The point (-5, -5) lies in Quadrant III.

Test Case 5
Input:
Enter the coordinates (x, y): 4 -1
Output:
The point (4, -1) lies in Quadrant IV.

Test Case 6
Input:
Enter the coordinates (x, y): 0 0
Output:
The point (0, 0) is at the origin.

Test Case 7
Input:
Enter the coordinates (x, y): 0 9
Output:
The point (0, 9) lies on the y-axis.

Test Case 8
Input:
Enter the coordinates (x, y): 5 0
Output:
The point (5, 0) lies on the x-axis.

Test Case 9
Input:
Enter the coordinates (x, y): -11 -14
Output:
The point (-11, -14) lies in Quadrant III

Test Case 10
Input:
Enter the coordinates (x, y): Two
Output:
Invalid input. Please enter valid numeric coordinates.