| Day 5 | Operators in C |
|---|---|

| | |
|---|---|
| **Objectives** | To apply operators effectively within conditional statements and loops to control program flow. |
| **Outcomes** | Students will be able to<br>1. Evaluate complex expressions accurately by understanding operator precedence and associativity.<br>2. Use operators effectively within conditional statements and loops to manage and control the flow of programs. |

**1. Subtraction of Two Numbers**

In a retail store, inventory management involves tracking the quantity of items in stock. When a customer purchases items, the store needs to update its inventory by subtracting the quantity sold from the current stock. Write a C program that should display the results of each operation.

**Constraints**

Positive Quantities: The current stock and quantities sold must be non-negative integers.

Valid Stock Check: The quantity sold for each item must not exceed the current stock of that item.

Termination on Error: If any sold quantity exceeds the current stock, the program should output an error message and terminate without updating the inventory.

**Input Range**:

Ø The program checks if the quantities sold (sold_apples, sold_oranges, sold_bananas) exceed the current stock (apples, oranges, bananas).

**Output Range**:

Ø If so, it prints an error message and terminates.

Test case 1:
Input:
Apples: 100
Oranges: 75
Bananas: 50
Sold apples: 0
Sold oranges: 0
Sold bananas: 0
Output:

Apples: 100
Oranges: 75
Bananas: 50

Test case 2:
Input:
Apples: 100
Oranges: 75
Bananas: 50
Sold apples: 0
Sold oranges: 80
Sold bananas: 0
Output:
Error: sold quantities exceed current stock. Please adjust.

Test case 3:
Input:
Apples: 100
Oranges: 75
Bananas: 50
Sold apples: 0
Sold oranges: 10
Sold bananas: 15
Output:
Apples: 100
Oranges: 65
Bananas: 35

Test Case 4:
Input:
Apples: 80
Oranges: 40
Bananas: 25
Sold Apples: 30
Sold Oranges: 10
Sold Bananas: 5
Output:
Apples: 50
Oranges: 30
Bananas: 20

Test Case 5:
Input:
Apples: 150
Oranges: 100
Bananas: 80
Sold Apples: 50
Sold Oranges: 100
Sold Bananas: 50
Output:
Apples: 100
Oranges: 0
Bananas: 30

Test Case 6:
Input:
Apples: 120
Oranges: 90
Bananas: 60
Sold Apples: 130
Sold Oranges: 10
Sold Bananas: 20
Output:
Error: Sold quantities exceed current stock. Please adjust.

Test Case 7:
Input:
Apples: 100
Oranges: 75
Bananas: 50
Sold Apples: 0
Sold Oranges: -10
Sold Bananas: 0
Output:
Error: Sold quantities cannot be negative. Please check

Test Case 8:
Input:
Apples: 100

Oranges: 50
Bananas: 25
Sold Apples: 105
Sold Oranges: 5
Sold Bananas: 10
Output:
Error: Sold quantities exceed current stock. Please adjust.

Test Case 9:
Input:
Apples: 50
Oranges: 30
Bananas: 10
Sold Apples: -20
Sold Oranges: -10
Sold Bananas: 0
Output:
Error: Sold quantities cannot be negative. Please check

Test Case 10:
Input:
Apples: 60
Oranges: 40
Bananas: 30
Sold Apples: 60
Sold Oranges: 5
Sold Bananas: 10
Output:
Apples: 0
Oranges: 35
Bananas: 20

**2. Division of Two Number**

In culinary arts, recipe scaling is a common practice when adjusting ingredient quantities to serve a different number of people or to fit different sized cooking vessels. Division plays a crucial role in determining how much of each ingredient is needed when scaling a recipe up or down. Write a C program that should display the results of each operation.

**Constraints:**

Number of Servings: Must be a positive integer greater than zero.

Non-Negative Outputs: The calculated quantities for flour, sugar, and milk should be non-negative and properly formatted to two decimal places.

**Input Range**:

Ø The program ensures that the number of servings (new_servings) entered by the user is greater than zero.

**Output Range**:

Ø The output quantities (scaled_flour, scaled_sugar, scaled_milk) are non-negative and represent valid measurements in cups.

Test Case 1:
Input:
Number of Servings: 8
Output:
Scaled Recipe for 8 Servings:
Flour: 4.00 cups
Sugar: 2.00 cups
Milk: 1.04 cups

Test Case 2:
Input:
Number of Servings: 10
Output:
Scaled Recipe for 10 Servings:
Flour: 5.00 cups
Sugar: 2.50 cups
Milk: 1.30 cups

Test Case 3:
Input:
Number of Servings: 1
Output:
Scaled Recipe for 1 Serving:
Flour: 0.50 cups
Sugar: 0.25 cups
Milk: 0.13 cups

Test Case 4:
Input:
Number of Servings: -10

Output:
ERROR!
Error: Number of servings must be greater than zero.

Test Case 5:
Input:
Number of Servings: 0
Output:
ERROR!
Error: Number of servings must be greater than zero.

Test Case 6:
Input:
Number of Servings: 15
Output:
Scaled Recipe for 15 Servings:
Flour: 7.50 cups
Sugar: 3.75 cups
Milk: 1.95 cups

Test Case 7:
Input:
Number of Servings: 5
Output:
Scaled Recipe for 5 Servings:
Flour: 2.50 cups
Sugar: 1.25 cups
Milk: 0.65 cups

Test Case 8:
Input:
Number of Servings: 20
Output:
Scaled Recipe for 20 Servings:
Flour: 10.00 cups
Sugar: 5.00 cups
Milk: 2.60 cups

Test Case 9:
Input:
Number of Servings: 7
Output:
Scaled Recipe for 7 Servings:
Flour: 3.50 cups
Sugar: 1.75 cups
Milk: 0.91 cups

Test Case 10:
Input:
Number of Servings: 25
Output:
Scaled Recipe for 25 Servings:
Flour: 12.50 cups
Sugar: 6.25 cups
Milk: 3.25 cups

**3**. **Pre-Increment Operator**

In an elevator control system, there might be multiple floors that the elevator can stop at. Each time the elevator reaches a floor, you want to update and display the current floor number. Here's how the pre-increment operator can be used: Write a C program to display the updated value of the variable using the pre-increment operator.

**Constraints:**

**Number of Floors:**

The total number of floors (num_floors) must be a positive integer, greater than 1.

**Target Floor Validity:**

The target floor entered by the user must be an integer within the range from 0 to num_floors -1.

**Starting Floor:**

The elevator starts at floor 0.

**Movement Limits:**

The elevator cannot move below floor 0 or above num_floors - 1.

**Pre-Increment Operation:**

Use the pre-increment operator (++current_floor) to update the current floor number before displaying it.

**Output Requirements:**

Display the current floor number in the format: "Elevator stops at Floor X" where X is the floor number.

**Example:**

For a building with 5 floors (numbered 0 through 4):
Valid Target Floors: 0, 1, 2, 3, 4
Invalid Target Floors: Any number less than 0 or greater than 4

Test Case 1:
Input:
Target Floor: -1
Output:
Invalid target floor. Please enter a value between 0 and 9.

Test Case 2:
Input:
Target Floor: 0
Output:
Elevator stays at Floor 0

Test Case 3:
Input:
Target Floor: 10
Output:
Invalid target floor. Please enter a value between 0 and 9.

Test Case 4:
Input:
Target Floor: 3
Output:
Elevator stops at Floor 1
Elevator stops at Floor 2
Elevator stops at Floor 3

Test Case 5:
Input:
Target Floor: 8
Output:
Elevator stops at Floor 1
Elevator stops at Floor 2
Elevator stops at Floor 3
Elevator stops at Floor 4

Elevator stops at Floor 5
Elevator stops at Floor 6
Elevator stops at Floor 7
Elevator stops at Floor 8

Test Case 6:
Input:
Target Floor: 7
Output:
Elevator stops at Floor 1
Elevator stops at Floor 2
Elevator stops at Floor 3
Elevator stops at Floor 4
Elevator stops at Floor 5
Elevator stops at Floor 6
Elevator stops at Floor 7

Test Case 7:
Input:
Target Floor: 2
Output:
Elevator stops at Floor 1
Elevator stops at Floor 2

Test Case 8:
Input:
Target Floor: 6
Output:
Elevator stops at Floor 1
Elevator stops at Floor 2
Elevator stops at Floor 3
Elevator stops at Floor 4
Elevator stops at Floor 5
Elevator stops at Floor 6

Test Case 9:
Input:
Target Floor: 9
Output:

Elevator stops at Floor 1
Elevator stops at Floor 2
Elevator stops at Floor 3
Elevator stops at Floor 4
Elevator stops at Floor 5
Elevator stops at Floor 6
Elevator stops at Floor 7
Elevator stops at Floor 8
Elevator stops at Floor 9

Test Case 10:
Input:
Target Floor: 4
Output:
Elevator stops at Floor 1
Elevator stops at Floor 2
Elevator stops at Floor 3
Elevator stops at Floor 4

**4.Demonstrate Logical AND operator:**

Develop a smart door lock system with two-factor authentication. The user must enter the correct PIN code and scan their fingerprint to unlock the door.The program should take two integer values as input, perform the logical AND operation on them, and display the result using the C program.

**Constraints**

Input Values: The values for a and b should be integers. They can be positive, negative, or zero.
Output: The result should be the logical AND of the two integers. The result will be 1 if both integers are non-zero, otherwise 0.
Logical AND Operation: In C, the logical AND operation is typically performed using the && operator. For integers, this means checking if both integers are non-zero.

Test case 1:
Input:
A = 5 ,b = 10
Output:
Logical and result: 1

Test case 2:
Input:
A = 0 ,b = 0
Output:
Logical and result: 0

Test case 3:
Input:
A = -1 ,b = 1
Output:
Logical and result: 1

Test case 4:
Input:
A = 3, b = 7
Output:
Logical and result: 1

Test case 5:
Input:
A = -5, b = -10
Output:
Logical and result: 1

Test case 6:
Input:
A = 0, b = 5
Output:
Logical and result: 0

Test case 7:
Input:
A = 5, b = -1
Output:
Logical and result: 1

Test case 8:
Input:
A = -3, b = 0

Output:
Logical and result: 0

Test case 9:
Input:
A = 0, b = 0
Output:
Logical and result: 0

Test case 10:
Input:
A = 0, b = -50
Output:
Logical and result: 0

**5. Smallest of Two Numbers Using Ternary Operator:**

Imagine you're developing a temperature monitoring system for a greenhouse or a data logger for environmental conditions. You have sensors that measure the temperature at two different points, and you need to determine which temperature reading is smaller.

**Constraints:**

Data Type: Temperatures are represented using the float data type to handle decimal values.
Precision: Temperatures are printed with one decimal place.
Use a ternary operator.·

**Output Range**:

For each valid test case, the program outputs:
The temperature readings from both sensors.
The smallest temperature detected using the ternary operator.

Test case 1:
Input:
Sensor 1 temperature: 24.5
Sensor 2 temperature: 22.8
Output:
Temperature sensor 1: 24.5°c
Temperature sensor 2: 22.8°c
Smallest temperature: 22.8°c

Test case 2:
Input:
Sensor 1 temperature: 10.5
Sensor 2 temperature: 15.2
Output:
Temperature sensor 1: 10.5°c
Temperature sensor 2:15.2°c
Smallest temperature: 10.5°c

Test case 3:
Input:
Sensor 1 temperature: -5.0
Sensor 2 temperature: -8.0
Output:
Temperature sensor 1: -5.0°c
Temperature sensor 2: -8.0°c
Smallest temperature: -8.0°c

Test case 4:
Input:
Sensor 1 temperature: 30.0
Sensor 2 temperature: 30.0
Output:
Temperature sensor 1: 30.0°c
Temperature sensor 2: 30.0°c
Smallest temperature: 30.0°c

Test case 5:
Input:
Sensor 1 temperature: 0.0
Sensor 2 temperature: -10.0
Output:
Temperature sensor 1: 0.0°c
Temperature sensor 2: -10.0°c
Smallest temperature: -10.0°c

Test case 6:
Input:
Sensor 1 temperature: 50.0

Sensor 2 temperature: 45.5
Output:
Temperature sensor 1: 50.0°c
Temperature sensor 2: 45.5°c
Smallest temperature: 45.5°c

Test case 7:
Input:
Sensor 1 temperature: -30.0
Sensor 2 temperature: -25.0
Output:
Temperature sensor 1: -30.0°c
Temperature sensor 2: -25.0°c
Smallest temperature: -30.0°c

Test case 8:
Input:
Sensor 1 temperature: 100.0
Sensor 2 temperature: 95.5
Output:
Temperature sensor 1: 100.0°c
Temperature sensor 2: 95.5°c
Smallest temperature: 95.5°c

Test case 9:
Input:
Sensor 1 temperature: 25.0
Sensor 2 temperature: -60.0
Output:
Temperature sensor 1: 25.0°c
Temperature sensor 2: -60.0°c
Smallest temperature: -60.0°c

Test case 10:
Input:
Sensor 1 temperature: 0.0
Sensor 2 temperature: 100.0
Output:
Temperature sensor 1: 0.0°c

Temperature sensor 2: 100.0°c
Smallest temperature: 0.0°c

**6. Modulus of Two Numbers**

In virtual memory systems, page tables map virtual addresses to physical memory addresses. A virtual address is divided into two parts: a page number and an offset within the page. Provided with a virtual address (virtualAddress) and the page size (pageSize), write a C program to use the modulus operation to extract the page offset (the remainder) from the virtual address.

**Constraints:**

Virtual Address Constraints:
Type: Integer
Range: 0 to 2,147,483,647 (standard int range in C)
Description: The virtual address should be a non-negative integer, as it represents a specific location within a memory space.
Page Size Constraints:
Type: Integer
Range: 1 to 2,147,483,647 (standard int range in C)
Description: The page size should be a positive integer. It represents the size of each memory page and must be greater than zero.

Test Case 1:
Input:
Enter the virtual address: 85
Enter the page size: 200
Output:
Page Offset: 85

Test Case 2:
Input:
Enter the virtual address: 99
Enter the page size: 20
Output:
Page Offset: 19

Test Case 3:
Input:
Enter the virtual address: 153
Enter the page size: 20

Output:
Page Offset: 13

Test Case 4:
Input:
Enter the virtual address: 2147483647
Enter the page size: 2147483647
Output:
Page Offset: 0

Test Case 5:
Input:
Enter the virtual address: 2147483648
Output:
ERROR!
Error: Virtual address must be between 0 and 2147483647.

Test Case 6:
Input:
Enter the virtual address: 3562
Enter the page size: 2147483649
Output:
ERROR!
Error: Page size must be a positive integer greater than zero and less than or equal to 2147483647.

Test Case 7:
Input:
Enter the virtual address: 0
Enter the page size: 0
Output:
ERROR!
Error: Page size must be a positive integer greater than zero and less than or equal to 2147483647.

Test Case 8:
Input:
Enter the virtual address: 0
Enter the page size: 2147483647

Output:
Page Offset: 0

Test Case 9:
Input:
Enter the virtual address: 310
Enter the page size: 75
Output:
Page Offset: 10

Test Case 10:
Input:
Enter the virtual address: 450
Enter the page size: 90
Output:
Page Offset: 0

**7. Greatest of Three Numbers Using Relational Operators**

A recruitment team for a company wants to conduct interviews for a position. Each candidate is evaluated based on three criteria: technical skills, communication skills, and experience and determines which candidate has the highest combined score to decide who should be selected for the job. Write a C program that should display the results of each operation.

**Constraints:**

**Input Range**:

Ø Each score (technicalSkills, communicationSkills, experience) should be an integer between 0 and 100, inclusive.

**Output Range**:

Ø The program outputs the integer for each test case, demonstrating the functionality of determining the highest score among the criteria.

Test case 1:
Input:
Enter the score for technical skills: 20
Enter the score for communication skills: 60
Enter the score for experience: 80
Output: 80

Test case 2:
Input:
Enter the score for technical skills: 95
Enter the score for communication skills: 85
Enter the score for experience: 60
Output: 95

Test case 3:
Input:
Enter the score for technical skills: 80
Enter the score for communication skills: 70
Enter the score for experience: 90
Output: 90

Test case 4:
Input:
Enter the score for technical skills: 60
Enter the score for communication skills: 60
Enter the score for experience: 60
Output: 60

Test case 5:
Input:
Enter the score for technical skills: 0
Enter the score for communication skills: 30
Enter the score for experience: 40
Output: 40

Test case 6:
Input:
Enter the score for technical skills: 0
Enter the score for communication skills: 50
Enter the score for experience: 0
Output: 50

Test case 7:
Input:
Enter the score for technical skills: 0
Enter the score for communication skills: 0

Enter the score for experience:
0
Output: 0

Test case 8:
Input:
Enter the score for technical skills: -75
Enter the score for communication skills: 20
Enter the score for experience: 60
Invalid input! Scores must be between 0 and 100.

Test case 9:
Input:
Enter the score for technical skills: *75
Enter the score for communication skills: enter the score for experience: invalid input! Scores must be between 0 and 100.

Test case 10:
Input:
Enter the score for technical skills: 60
Enter the score for communication skills: 58
Enter the score for experience: -30
Invalid input! Scores must be between 0 and 100.

**8**. **Demonstrate Logical OR operator**

Develope an online shopping application where customers can purchase items. You need to validate whether a customer's shopping cart meets certain conditions to qualify for free shipping or a discount. When the total purchase amount is greater than 999 dollars or total items in cart are greater than 5 , then he is eligible for free shipping.

**Constraints:**

**Input Range**:

Ø 'cartTotal' should be an integer between 0 and 1000 (assuming dollars).
Ø 'numberOfItems' should be an integer between 0 and 20.

Test Case 1:
Input:
Enter the cart total in Dollars: 15
Enter the number of items : 6
Output:
Customer qualifies for free shipping or discount.

Test Case 2:
Input:
Enter the cart total in Dollars: 80
Enter the number of items : 6
Output:
Customer qualifies for free shipping or discount.

Test Case 3:
Input:
Enter the cart total in Dollars: 90
Enter the number of items : 4
Output:
Customer does not qualify.

Test Case 4:
Input:
Enter the cart total in Dollars: 100
Enter the number of items : 2
Output:
Customer qualifies for free shipping or discount.

Test Case 5:
Input:
Enter the cart total in Dollars: 1111
Enter the number of items : 5
Output:
Invalid input! Please enter valid values.

Test Case 6:
Input:
Enter the cart total in Dollars: 999
Enter the number of items : -2
Output:
Invalid input! Please enter valid values.

Test Case 7:
Input:
Enter the cart total in Dollars: 1111

Enter the number of items : -2
Output:
Invalid input! Please enter valid values.

Test Case 8:
Input:
Enter the cart total in Dollars: 90
Enter the number of items : 6
Output:
Customer qualifies for free shipping or discount.

Test Case 9:
Input:
Enter the cart total in Dollars: 0
Enter the number of items : 0
Output:
Customer does not qualify.

Test Case 10:
Input:
Enter the cart total in Dollars: 1000
Enter the number of items : 20
Output:
Customer qualifies for free shipping or discount.

**9. Currency converter using division and modulo division operator:**

A financial system needs to represent monetary values internally as the smallest unit of currency (cents) for accuracy. When displaying amounts to users, the system must convert these cent values into a human-readable dollar and cent format.

**Constraints:**

**Input Range**:

Ø The cents input should be a non-negative integer.

**Output Range**:

Ø The output dollars should be an integer that represents the equivalent amount in dollars and cents.

Test case 1:
Input:
Enter the amount in cents: 250

Output:
250 cents is equal to 2 dollars and 50 cents

Test case 2:
Input:
Enter the amount in cents: 128
Output:
128 cents is equal to 1 dollars and 28 cents

Test case 3:
Input:
Enter the amount in cents: 72
Output:
72 cents is equal to 0 dollars and 72 cents

Test Case 4:
Input:
Enter the amount in cents: 500
Output:
500 cents is equal to 5 dollars and 0 cents

Test Case 5:
Input:
Enter the amount in cents: 0
Output:
Invalid currency

Test Case 6:
Input:
Enter the amount in cents: -50
Output:
Invalid currency

Test Case 7:
Input:
Enter the amount in cents: 1000
Output:
1000 cents is equal to 10 dollars and 0 cents

Test Case 8:
Input:
Enter the amount in cents: -200
Output:
Invalid currency

Test Case 9:
Input:
Enter the amount in cents: 432
Output:
432 cents is equal to 4 dollars and 32 cents

Test Case 10:
Input:
Enter the amount in cents: 800
Output:
800 cents is equal to 8 dollars and 0 cents

**10. Greatest of three numbers using Ternary Operator**

Imagine you are developing a program to determine the highest grade between three test scores for a student. You want to use the ternary operator to efficiently compare the scores and determine the greatest score.

**Constraints:**

**Input Range**:

Ø The numbers a,b and c should be positive integers within a specified range(0 to 100).

Test case 1:
Input:
Score 1: 10
Score 2: 20
Score 3: 30
Output:
Highest score: 30

Test case 2:
Input:
Score 1: 99
Score 2: 95
Score 3: 85

Output:
Highest score: 99

Test case 3:
Input:
Score 1: 95
Score 2: 95
Score 3: 95
Output:
Highest score: 95

Test case 4:
Input:
Score 1: 99
Score 2: 95
Score 3: 85
Output:
Highest score: 99

Test case 5:
Input:
Score 1: -80
Score 2: 20
Score 3: 50
Output:
Scores cannot be negative. Please check

Test case 6:
Input:
Score 1: -50
Score 2: -30
Score 3: -50
Output:
Scores cannot be negative. Please check

Test case 7:
Input:
Score 1: -50
Score 2: 80

Score 3: -60
Output:
Scores cannot be negative. Please check

Test case 8:
Input:
Score 1: 80
Score 2: -70
Score 3: 90
Output:
Scores cannot be negative. Please check

Test case 9:
Input:
Score 1: 20
Score 2: -0
Score 3: 30
Output:
Highest score: 30

Test case 10:
Input:
Score 1: -0
Score 2: -0
Score 3: -0
Output:
Highest score: 0