Activity 1: Who Are You?

Problem:

A school registration system needs to collect and display a student's full name using separate inputs for first and last names.

Acceptance Criteria:

- User inputs first name and last name as strings.
- Output format: Your full name is: Firstname Lastname.
- Reject empty input fields.

Activity 2: How Old Are You in Months?

Problem:

A pediatric clinic tracks patient age in months. They need a program that converts a user's age in years to months.

Acceptance Criteria:

- Accept an integer input for age.
- Output format: You are {age_in_months} months old.
- Age must be between 1 and 120.

Activity 3: Can You Afford It?

Problem:

A store needs to compute the total cost of an item based on its price and quantity purchased.

Acceptance Criteria:

- Accept price as float and quantity as integer.
- Output format: Total cost: Php {total} (2 decimal places).
- Both inputs must be positive.

Activity 4: Feeling Hot or Cold?

Problem:

A weather app wants to convert a temperature entered in Celsius to Fahrenheit using the formula: $F = (C \times 9/5) + 32$.

Acceptance Criteria:

- Input must be a number (float or int).
- Output: Temperature in Fahrenheit: {value}.
- Input must be between -100 and 100.

Activity 5: Who Are You, Really?

Problem:
An online form captures a user's name, age, and height, and summarizes it in one sentence.
Acceptance Criteria:

- Declare or input: name (string), age (int), height (float).
- Output: Name: John, Age: 25, Height: 5.9
- All data must be valid and properly typed.

Activity 6: Did You Pass?
Problem:
A teacher wants to enter a student's numeric grade and display a corresponding letter grade with validation.
Acceptance Criteria:

- Grade input must be integer from 0 to 100.
- Output letter grades:
  A = 90–100
  B = 80–89
  C = 70–79
  D = 60–69
  F = <60
- Display: Your grade is: B (for example).
- Reject grades outside 0–100.

Activity 7: The Simple Calculator
Problem:
A utility app allows users to enter two numbers and a math operation to calculate the result.
Acceptance Criteria:

- Accept two float inputs and one operator (+, -, *, /).
- Output: The result is: {result}.
- Prevent division by zero.

Activity 8: Convert Me!
Problem:
A data entry tool receives numeric input as a string and needs to convert it into an integer before performing calculations.
Acceptance Criteria:

- Input is a string that must contain only digits.

- Convert to integer and add 10.
- Output: Result after adding 10: {value}.
- Reject non-numeric input.

Activity 9: Even or Odd?

Problem:

An analytics system wants to check whether a given number is even or odd to determine data routing.

Acceptance Criteria:

- Input must be an integer.
- Output must be:
  The number is even. or
  The number is odd.
- Reject invalid or non-integer input.

Activity 10: Validate My Info

Problem:

An online profile system must verify if a user's name, age, and email address are valid before submission.

Acceptance Criteria:

- Name must not be empty.
- Age must be an integer between 1 and 120.
- Email must contain '@'.
- Output:
  - All fields are valid. or
  - Invalid age. / Invalid email. / Name required.

Activity 11: PIN Code Retry System

Problem:

An ATM allows a maximum of 3 attempts to enter the correct 4-digit PIN ("1234"). If the user fails all attempts, access is denied.

Acceptance Criteria:

- Input: PIN as a string.
- Loop: Maximum of 3 tries.
- Output:

- "Access granted" if correct.
- "Access denied" after 3 failures.

Activity 12: Even or Odd Number Checker

Problem:
The program asks the user to input a number and determines if it's even or odd using modulus and conditional statements.

Acceptance Criteria:

- Input: integer.
- Output: "Even number" or "Odd number".
- Must reject non-integer values.

Activity 13: Name Case Formatter

Problem:
Ask for a name input and return it with the first letter in uppercase and the rest in lowercase (e.g., john → John).

Acceptance Criteria:

- Input: string (name).
- Output: formatted name.
- Validate that input is alphabetic and not empty.

Activity 14: Age Group Categorizer

Problem:
Based on age input, classify the user:

- 0–12 → Child
- 13–19 → Teen
- 20–59 → Adult

- 60+ → Senior

Acceptance Criteria:

- Input: integer (age).
- Output: corresponding category.
- Validate age to be between 1–130.

Activity 15: Letter Grade Calculator

Problem:
Ask for a numeric grade (0–100) and determine its equivalent letter grade using a switch or if-else.

Acceptance Criteria:

- Input: integer grade.
- Output:
    - 90–100: A
    - 80–89: B
    - 70–79: C
    - 60–69: D
    - <60: F
- Reject values outside 0–100.

Activity 16: Simple Calculator with Switch

Problem:
Create a calculator that performs +, -, *, or / based on user-selected operator.

Acceptance Criteria:

- Inputs: two floats, one operator as a string.
- Output: result based on operator.
- Use switch or if-else.

- Validate division by zero.

Activity 17: Multiplication Table Printer

Problem:
Ask for a number and print its multiplication table up to 10 using a for loop.

Acceptance Criteria:

- Input: integer 1–10.
- Output: formatted table.
- Validate input range.

Activity 18: Countdown Timer

Problem:
Let the user input a number and count down to 0 using a while loop.

Acceptance Criteria:

- Input: integer > 0.
- Output: numbers decreasing by 1 until 0.
- Validate input is positive.

Activity 19: Secret Word Guesser

Problem:
Ask the user to guess the secret word ("open"). Use a do-while loop until the correct word is entered.

Acceptance Criteria:

- Input: word (case-insensitive).
- Output: "Try again" until matched.
- End with "Correct!".

Activity 20: Password Policy Validator

Problem:
Ask for a password. Accept only if it's ≥8 characters, includes a number, and a capital letter.

Acceptance Criteria:

- Input: string (password).
- Output: "Valid password" or list missing rules.
- Use string length, char.IsDigit(), and char.IsUpper().

Activity 21: User Menu with Switch

Problem:
Show a menu with 3 options: [1] Greet, [2] Show Date, [3] Exit. Use switch.

Acceptance Criteria:

- Input: choice as integer.
- Output:
  - 1 → "Hello, User!"
  - 2 → Show current date
  - 3 → "Exiting..."
- Handle invalid options.

Activity 22: String Character Access

Problem:
Ask user to input a word and a position (starting from 0). Show the character at that position.

Acceptance Criteria:

- Input: word and index.
- Output: character at given index.

- Validate that index is within range.

Activity 23: Uppercase Letter Counter

Problem:
Ask the user to input a sentence. Count and display how many uppercase letters are present.

Acceptance Criteria:

- Input: sentence string.
- Output: total uppercase letters.
- Use char.IsUpper() in a loop.

Activity 24: Simple Authentication Loop

Problem:
User enters a username and password. Repeat until both match predefined values.

Acceptance Criteria:

- Hardcoded username: "admin", password: "1234"
- Loop until both match.
- Case-sensitive.
- Display "Login successful" or "Try again".

Activity 25: Total Until Stop Word

Problem:
Keep asking the user to enter numbers and sum them until the user types "stop".

Acceptance Criteria:

- Inputs: string (must be converted to int).
- Stop if input is "stop" (case-insensitive).
- Output: sum of all valid numbers.

- Ignore invalid (non-numeric) entries.

Activity 26: Longest Word Length

Problem:
Ask the user to enter a sentence. Determine and print the length of the **longest word** using only loops (no .Split() or arrays).

Acceptance Criteria:

- Input: a sentence string
- Output: length of the longest word
- Words are separated by one space
- No built-in string split or arrays used

Activity 27: Number Pyramid Generator

Problem:
Write a program that displays a pyramid of consecutive numbers using nested loops.
Input: 4
Output:

1

2 3

4 5 6

7 8 9 10

Acceptance Criteria:

- Input: number of rows
- Output: number triangle
- No arrays or string lists used
- Proper use of nested for or while loops

Activity 28: Manual Word Reversal

Problem:
Ask the user to input a word and display it in reverse using loop and string indexing.

Acceptance Criteria:

- Input: single word
- Output: reversed word
- No built-in reverse or array methods
- Use loop and character access only

Activity 29: Prime Numbers in Range

Problem:
Let the user enter a start and end value, then list all **prime numbers** between them.

Acceptance Criteria:

- Input: start and end integers
- Output: prime numbers in the range
- Use nested loops and modulus
- Validate range (start < end)

Activity 30: Numbers to Words (0–999)

Problem:
Convert a number between 0 and 999 into **English words**.
e.g., 123 → One Hundred Twenty Three

Acceptance Criteria:

- Input: integer (0–999)
- Output: English phrase
- Use only conditionals and switch
- No arrays or dictionaries

Activity 31: Palindrome Word Checker

Problem:
Check if a word is a **palindrome** (reads the same backward).

Acceptance Criteria:

- Input: single word
- Output: "Palindrome" or "Not a palindrome"
- Use loop and char comparison (no reverse)

Activity 32: Alphabet Ladder Printer

Problem:
Let the user input a letter (A–Z). Display a pattern ladder from A to the input letter.
E.g., Input: D
Output:

A

AB

ABC

ABCD

Acceptance Criteria:

- Input: capital letter A–Z
- Output: ladder pattern
- Use ASCII arithmetic and nested loops

Activity 33: Manual Upper/Lower Case Converter

Problem:
Ask the user for a sentence and allow them to convert it to either uppercase or lowercase **without using .ToUpper() or .ToLower()**.

Acceptance Criteria:

- Input: sentence and choice ("upper"/"lower")
- Output: converted sentence
- Use ASCII manipulation and loops

Activity 34: Digital Root Calculator

Problem:
Continuously sum the digits of a number until only one digit remains.
e.g., 456 → 15 → 6

Acceptance Criteria:

- Input: positive integer
- Output: single-digit result
- Use loops and modulus/division only
- No arrays, recursion, or string parsing

Activity 35: Title Case Formatter

Problem:
Convert a sentence so that the first letter of each word is capitalized, and the rest are lowercase.
Input: "hELLO wORld" → Output: "Hello World"

Acceptance Criteria:

- Input: sentence
- Output: formatted sentence
- Use loops, char casing logic, and position checks
- No arrays or built-in case transformation functions allowed