**CS 201R**

**Problem Solving & Programming II**

**Program 4 – Calculator Due \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**C++ Calculator Assignment: Number Conversion & Basic Operations**

Objective: To create a console-based C++ calculator application that can perform basic arithmetic operations (addition, subtraction, multiplication, division) and number base conversions (decimal, binary, octal, hexadecimal).

Learning Outcomes:

* Reinforce understanding of C++ data types, input/output, and control structures (if-else, switch, loops).
* Implement functions for modularity and reusability.
* Gain experience with string manipulation and numerical conversions.
* Develop problem-solving skills in handling different number bases.

Requirements:

1. Menu-Driven Interface:
   * The program should present a clear, user-friendly menu with options for:
     + Arithmetic Operations
     + Number Base Conversions
     + Exit
2. Arithmetic Operations Module:
   * When selected, this module should prompt the user to choose an operation (add, subtract, multiply, divide).
   * It should then prompt the user to enter two numbers (integers or floating-point, your choice, but clearly state which).
   * Perform the chosen operation and display the result.
   * Handle division by zero gracefully.
   * After an operation, offer the user the option to perform another arithmetic operation or return to the main menu.
3. Number Base Conversion Module:
   * When selected, this module should present a sub-menu for conversions:
     + Decimal to Binary
     + Decimal to Octal
     + Decimal to Hexadecimal
     + Binary to Decimal
     + Octal to Decimal
     + Hexadecimal to Decimal
     + (Optional but recommended: any-to-any conversion if you're feeling ambitious!)
   * For each conversion, prompt the user to enter the number in the specified base.
   * Validate the input to ensure it's a valid number for the given base (e.g., binary numbers only contain '0' and '1').
   * Perform the conversion and display the result.
   * After a conversion, offer the user the option to perform another conversion or return to the main menu.
4. Error Handling & Input Validation:
   * Handle invalid menu choices (e.g., user enters '5' when there are only 3 options).
   * Validate numerical inputs to prevent crashes (e.g., non-numeric input when a number is expected).
   * Ensure proper input for number base conversions (e.g., only '0' and '1' for binary).
5. Functions:
   * Break down the program into well-defined functions. Examples include:
     + displayMainMenu()
     + performArithmeticOperations()
     + performNumberConversions()
     + add(num1, num2)
     + subtract(num1, num2)
     + multiply(num1, num2)
     + divide(num1, num2)
     + decToBin(decimalNum)
     + decToOct(decimalNum)
     + decToHex(decimalNum)
     + binToDec(binaryNumString)
     + octToDec(octalNumString)
     + hexToDec(hexNumString)
     + (and any helper functions you deem necessary)
6. Looping & Program Flow:
   * The main program loop should continue until the user explicitly chooses to exit.
   * Sub-modules should also allow repeated operations until the user chooses to return to the main menu.

Implementation Notes & Hints:

* Number Base Conversions:
  + Decimal to Other Bases: Use modulo and division operations. For hexadecimal, you'll need to handle digits A-F.
  + Other Bases to Decimal: Use powers of the base. For hexadecimal, you'll need to convert A-F to their decimal equivalents (10-15).
  + String Manipulation: For binary, octal, and hexadecimal input, it's often easiest to read them as strings and then parse them character by character.
  + Consider using string and functions like stoi (C++11 and later) or custom parsing for converting strings to numbers and vice-versa.
  + For hexadecimal, you might find stringstream useful, or you can implement your own logic for converting characters '0'-'9', 'A'-'F' (or 'a'-'f') to their integer values.
* Input/Output: Use cout for output and cin for input. Remember to include <iostream>. For string manipulation, include <string>. For mathematical functions (like pow for conversions), include <cmath>.
* Clear Prompts: Always provide clear instructions to the user about what input is expected.

**Output sample:**

A screenshot of a computer

AI-generated content may be incorrect. **A computer screen with white text

AI-generated content may be incorrect.**

**Submission:**

* Turn in your code by providing the GitHub link in Canvas.
* Make sure to include design / pseudocode notes prior to each.