**Name: \_\_\_\_Mark Okin, Ryan Quinn\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CSC 366**

**Assignment #3 – Our Imaginary Machine Language (IML)**

50 points

**Assigned**: 02/27/2024

**Due:** 03/05/2024

**Evaluation:**

1. Program a (\_\_\_\_\_ / 25)
2. Program b (\_\_\_\_\_ / 25)

**ASSIGNMENT**:

You may choose to work in pairs for this assignment.

In this assignment, you will become acquainted with machine language instructions and their simulated execution. You will be developing a virtual machine, and the “language” used will be the Imaginary Machine Language (IML). Here is a description of our simple virtual machine and IML. A Java implementation of this virtual machine can be found on Canvas:

* The virtual machine includes four data registers (R1, R2, R3, and R4).
* All information is handled in terms of words and a word is a signed six-digit decimal number.
* The machine contains a 100-word memory, referenced by location numbers 00,01, …, 99.
* Instructions are signed six-digit decimal numbers; thus they occupy one word of memory.
* The sign of an IML instruction is always positive but the sign of a data word may be either positive or negative.
* Each location in memory may contain an instruction, a data value, or an unused (and thus undefined) word.
* The last four digits of an IML instruction are the operands.
  + - Each operand is two decimal digits. An operand may be an address in memory or a register. Addresses in memory are 00-99. Registers are 01 through 04.
    - You will know from the type of instruction whether the operand refers to memory or a register.
    - The 3rd and 4th digits represent the first operand (either a memory address or a register).
    - The 5th and 6th digits represent the second operand (either a memory address or a register).
* The first two digits of each IML instruction are the operation code (opcode). The operations supported by IML include
  + READ = 12; Read a word from the keyboard into a specified memory location. The first operand is the memory location. The second operand is unused.
  + WRITE = 11; Write a word from a specific location to the screen. The first operand is the memory location. The second operand is unused.
  + LOAD = 22; Load a word from a specific location in memory into a specific register. The first operand is a register. The second operand is the memory location.
  + STORE = 21; Store the word from a specific register into a specific location in memory. The first operand is the register. The second operand is the memory location.
  + ADD = 31; Add the values in two registers. Leave the result in the first register listed. Both operands are registers.
  + SUBTRACT = 32; Start with the value in the first register listed and subtract the value in the second register listed. Leave the result in the first register. Both operands are registers.
  + MULTIPLY = 33; Multiply the values in two specific registers. Leave the result in the first register. Both operands are registers.
  + DIVIDE = 34; Start with the value in the first register listed and divide by the value in the second register listed. Leave the result in the first register. Both operands are registers.
  + BRANCH = 41; Jump to a specific location in memory. The first operand is a memory location. The second operand is unused. (Set the program counter to the specified memory location.)
  + BRANCHZERO = 42; Jump to a specific location in memory if a specific register contains zero. The first operand is a memory location. The second operand is a register.
  + BRANCHPOS = 43; Jump to a specific location in memory if a specific memory location contains a value greater than zero. The first operand is a memory location. The second operand is a register.
  + HALT = 99; Stop. No operands.

Your assignment is to do the following:

Write (in text files) IML programs to accomplish each of the following:

* 1. Write a program that will take two numbers input from the user and prints the smallest.
  2. Write a program that will determine the sum of a series of numbers that are entered. Accept input until the user enters 9999, which will indicate the end of input. Do not include 9999 in the sum. Print the sum.

Your code will be run on the Java implementation:

* 25pts: works correctly
* 23pts: runs with a minor error
* 20pts: runs with a major error
* 15pts: does not run, or crashes on normal input
* 0pts: no submission