Assignment 4
Fall 2019

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING
INDIANA UNIVERSITY

CSCI-C291 (System Programming with C & UNIX)

Submission Requirements:
Please complete all exercises and properly test them on silo. Submit *.c files for each coding exercise (and txt file for others) on Canvas (You may also submit the coding files on github in hw4 folder and other txt files on Canvas). The Source Code must include your name, date, and appropriate *comments* and ensure that the code is *properly organized* and *indented*. Make sure that the variable names are *descriptive* and the code is "readable".
Note: *Here is one way you can transfer files from your local computer to silo and back.*

Grading Scheme: Q1 (10), Q2 (7), Q3 (4), Q4 (34)
Note: 10% grade of each program will be assigned to: Proper Comments in code, Proper organization and indentation of the code, Code Readability i.e., proper names used for functions and variables

Coding Assignments:

1. Answer each of the following. Assume that unsigned integers are stored in 6 bytes and that the starting address of the array is at location 0x100400 in memory.
   a) Define an array of type `unsigned short` called `values` with ten elements, and initialize the elements to the even integers selected from the range between `2` and `20`. Assume the symbolic constant `SIZE` has been defined as `10`.
   b) Define a pointer `vPtr` that points to an object of type `unsigned short`.
   c) Print the elements of array `values` using array subscript notation. Use a `for` statement and assume integer control variable `i` has been defined.
   d) Give two separate statements that assign the starting address of array `values` to pointer variable `vPtr`.
   e) Print the elements of array `values` using pointer/offset notation.
   f) Print the elements of array `values` using pointer/offset notation with the array name as the pointer.
   g) Print the elements of array `values` by subscripting the pointer to the array.
   h) Refer to 5th element of array `values` using array subscript notation, pointer/offset notation with the array name as the pointer, pointer subscript notation, and pointer/offset notation.
   i) What address is referenced by `vPtr + 5`? What value is stored at that location?
   j) Assuming `vPtr` points to `values[6]`, what address is referenced by `vPtr -= 3`? What value is stored at that location?
2. For each of the following, write a single statement that performs the indicated task. Assume that long integer variables `value1` and `value2` have been defined and that `value1` has been initialized to decimal `70000`.
   a) Define the variable `lPtr` to be a pointer to an object of type `long int`.
   b) Assign the address of variable `value1` to pointer variable `lPtr`.
   c) Print the value of the object pointed to by `lPtr`.
   d) Assign the value of the object pointed to by `lPtr` to variable `value2`.
   e) Print the value of `value2`.
   f) Print the address of `value1`.
   g) Print the address stored in `lPtr`. Is the value printed the same as the address of `value1`?
3. Do each of the following:
   a) Write the function header for function `zero`, which takes a long integer array parameter `bigIntegers` and does not return a value.

b) Write the function prototype for the function in part (a).
c) Write the function header for function `add1AndSum`, which takes an integer array parameter `oneTooSmall` and returns an integer.
d) Write the function prototype for the function described in part (c).

4. ***(Modifications to the Simpletron Simulator)*** In Exercise 7.28 (*review attached pdf file*), you wrote a software simulation of a computer that executes programs written in Simpletron Machine Language (SML). In this exercise, we propose several modifications and enhancements to the Simpletron Simulator. We want to propose building a compiler that converts programs written in a high-level programming language (a variation of BASIC) to Simpletron Machine Language. Some of the following modifications and enhancements may be required to execute the programs produced by the compiler.

a) Extend the Simpletron Simulator's memory to contain 800 memory locations to enable the Simpletron to handle larger programs.
b) Allow the simulator to perform remainder calculations. This requires an additional Simpletron Machine Language instruction.
c) Allow the simulator to perform exponentiation calculations. This requires an additional Simpletron Machine Language instruction.
d) Modify the simulator to process floating-point values in addition to integer values.
e) Modify the simulator to allow output of SQRT value. This requires an additional Simpletron Machine Language instruction.
f) Modify the simulator to allow output of FACT value which would be the factorial of given number. This requires an additional Simpletron Machine Language instruction.
g) Modify the simulator to handle string input. [*Hint:* Each Simpletron word can be divided into two groups, each holding a two-digit integer. Each two-digit integer represents the ASCII decimal equivalent of a character. Add a machine-language instruction that will input a string and store it beginning at a specific Simpletron memory location. The first half of the word at that location will be a count of the number of characters in the string (i.e., the length of the string). Each succeeding half word contains one ASCII character expressed as two decimal digits. The machine-language instruction converts each character into its ASCII equivalent and assigns it to a half word.]