**Lab #8 Key: Pseudocode and Scratch Programming**

Due: Tuesday, April 1, 2014 beginning of class

## Instructions

* You will need your WFU-issued ThinkPad for this lab. This lab makes use of a programming language named *Scratch.* You should already have Scratch on your computer from the last lab. If not, download Scratch using the instructions under Resources -> Scratch in Sakai.
* Submit your lab report to the assignment in Sakai using this document and any other material you are asked to submit in the questions, e.g., a Scratch program.

NOTE TO GRADER: When checking the tables in the problems where students need to follow the code, they should get full credit for being correct. However, if they are on the right track but get off a little, this will throw off the entire table. Try to see if they know what they are doing and just got a little confused or if they totally don’t understand. Assign points accordingly. Also, sometimes they may repeat the same value for a variable in a row. This happens if the variable gets assigned the same value in the code. Some may choose to repeat the value in the row or simple leave as is. Either way is fine.

There are 5 questions, so just assign 20 points to each and split evenly among the parts.

## Part 0: Background

The purpose of this lab is to reinforce what you have learned about algorithms and pseudocode by translating them into a high-level computer language. Background comes from Chapters 6 and 7 (Computer Science Illuminated) and from lectures. For our purposes, the idea of programming is to take an algorithm in some form and translate it into *code that a computer can understand*. The computer in this case is the “compute agent” for the algorithm, much in the same way you might think of a chef as the compute agent for a recipe. We will use an excellent, introductory language, Scratch, that allows focus on problem solving and creativity rather than syntax by providing a graphical interface with drag-and-drop program components (statements). You will immediately see the ease of this approach as compared to machine or assembly code. However, you should remember that eventually your programs must be translated into machine code in order to run on your computer (not by you ☺).

When learning to program, particularly at this stage of experience, it is informative to both analyze and synthesize code. That is, read and study what others have done then write some of your own. You will be asked to describe what existing code is doing, modify code, and write your own.

## Part 1:

For this part you will work with the pseudocode that we discussed in class and will be asked to trace code that is reasonably complex for novices.

Note: In the code, you may see comments. These are denoted by the following: // “comments”. A double slash is often used to denote comments in pseudocode (as well as in some actual computer language code). Comments are there only for documentation and do not affect the code at all. They are there to help programmers understand what the code is doing.

**Question 1.**

**1.a.** Identify each variable in the pseudocode below, and enter its name in the left column of the table. ***Note: You may find it helpful to open a new window in Word so that you can show the table and the code at the same time.***

**//Ask the user for the number of values to process.**

**Write “How many values?”**

**Read length**

**Set counter to 0**

**WHILE (counter < length)**

**Read data[counter]**

**Set counter to counter + 1**

**Write “Enter value for which to search”**

**Read item**

**Set first to 0**

**Set last to length-1**

**Set found to FALSE**

**WHILE (first <= last AND NOT found)**

**Set middle to (first + last)/ 2**

**IF (item equals data[middle]))**

**Set found to TRUE**

**ELSE**

**IF (item < data[middle])**

**Set last to middle – 1**

**ELSE**

**Set first to middle + 1**

**Print found**

**1.b.** Assume the following values are input to the program (11, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 27). Complete the table by filling in the variable names and record their values as they change. **Each time a variable changes values, you are to enter its value in the next empty column for that variable. In this way, the last value for a variable is its current value. *Note: This is a change from the method described in class; however, after considerable discussion with others, this looks to be the most easily understood and will be the method used on the test.***

* If you need more rows in the table, please add them.
* If you need more columns, add another copy of the table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Variable Value** | | | | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| length | 11 |  |  |  |  |  |  |  |  |  |  |  |
| counter | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| item | 27 |  |  |  |  |  |  |  |  |  |  |  |
| first | 0 | 6 |  |  |  |  |  |  |  |  |  |  |
| last | 10 |  |  |  |  |  |  |  |  |  |  |  |
| found | F | T |  |  |  |  |  |  |  |  |  |  |
| middle | 5 | 8 |  |  |  |  |  |  |  |  |  |  |
| data[0] | 3 |  |  |  |  |  |  |  |  |  |  |  |
| data[1] | 6 |  |  |  |  |  |  |  |  |  |  |  |
| data[2] | 9 |  |  |  |  |  |  |  |  |  |  |  |
| data[3] | 12 |  |  |  |  |  |  |  |  |  |  |  |
| data[4] | 15 |  |  |  |  |  |  |  |  |  |  |  |
| data[5] | 18 |  |  |  |  |  |  |  |  |  |  |  |
| data[6] | 21 |  |  |  |  |  |  |  |  |  |  |  |
| data[7] | 24 |  |  |  |  |  |  |  |  |  |  |  |
| data[8] | 27 |  |  |  |  |  |  |  |  |  |  |  |
| data[9] | 30 |  |  |  |  |  |  |  |  |  |  |  |
| data[10] | 33 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**1.c.** What does this program do?

Search for 27 in an array

**1.d.** What algorithm from the book does this pseudocode implement?

Binary Search

**Question 2.**

**2.a.** Identify each variable in the pseudocode below, and enter its name in the left column of the table.

**//Ask the user to input the number of values she/he wants to use.**

**Write “How many values?”**

**Read length**

**Set counter to 0**

**WHILE (counter < length)**

**Read data[counter]**

**Set counter to counter + 1**

**Set current to 1**

**WHILE (current < length)**

**Set index to current**

**Set placeFound to FALSE**

**WHILE (index > 0 AND NOT placeFound)**

**IF (data[index] < data[index – 1])**

**Set tempItem to data[index]**

**Set data[index] to data[index - 1]**

**Set data[index - 1] to tempItem**

**Set index to index – 1**

**ELSE**

**Set placeFound to TRUE**

**Set current to current + 1**

**2.b.** Assume the following values are input to the program (4, Regina, Sarah, Harris, Jackson). Complete the table by filling in the variable names and record their values as they change. (See Question 1 for explanation.) If you need more rows in the table, please add them.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Variable Value** | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| length | 4 |  |  |  |  |  |  |  |  |
| counter | 0 | 1 | 2 | 3 | 4 |  |  |  |  |
| current | 1 | 2 | 3 | 4 |  |  |  |  |  |
| index | 1 | 2 | 1 | 0 | 3 | 2 | 1 |  |  |
| placeFound | F | T | F | F | T |  |  |  |  |
| tempItem | Harris | Harris | Jackson | Jackson |  |  |  |  |  |
| data[0] | Regina |  |  |  |  |  |  |  |  |
| data[1] | Sarah |  |  |  |  |  |  |  |  |
| data[2] | Harris |  |  |  |  |  |  |  |  |
| data[3] | Jackson |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**2.c.** What does this program do?

Sort the names in the array

**2.d.** What algorithm from the book does this pseudocode implement?

Insertion Sort

**Question 3.**

**3.a.** The following pseudocode computes the factorial of a number. Identify each variable in the pseudocode below, and enter its name in the left column of the table.

**Write “Enter an integer greater than or equal to zero”**

**Read number**

**Set index to number**

**Set value to 1**

**WHILE (index > 0)**

**value = value \* index**

**Set index to index – 1**

**Print “The final value is ”, value**

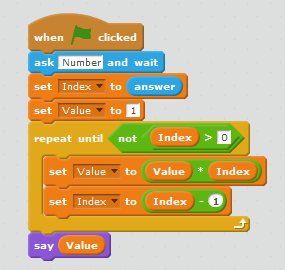
**3.b.** Assume the following value is input to the program (6). Complete the table by filling in the variable names and record their values as they change. (See Question 1 for explanation.) If you need more rows in the table, please add them.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Variable Value** | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| number | 6 |  |  |  |  |  |  |  |  |
| value | 1 | 6 | 30 | 120 | 360 | 720 |  |  |  |
| index | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |  |
|  |  |  |  |  |  |  |  |  |  |
| output | 720 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**3.c.** Do the following:

1. Write a Scratch program that implements the Pseudocode for this question.
2. What is the output of your program for the following inputs:

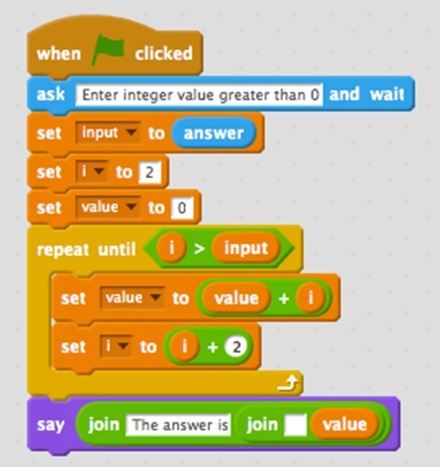
|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | **1** |
| **2** | **2** |
| **3** | **6** |
| **4** | **24** |
| **5** | **120** |
| **6** | **720** |



1. Save the program using the name *Lab8\_factorial.sb2*.
2. Submit the program to Sakai.

**Question 4.**

**4.a.** Identify each variable in the Scratch code below, and enter its name in the left column of the table.



**4.b.** Assume the following value is input to the program (7). Complete the table by filling in the variable names and record their values as they change. (See Question 1 for explanation.) If you need more rows in the table, please add them.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Variable Value** | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| input | 7 |  |  |  |  |  |  |  |  |
| i | 2 | 4 | 6 | 8 |  |  |  |  |  |
| value | 0 | 2 | 6 | 12 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| output | 12 |  |  |  |  |  |  |  |  |

**4.c.** For each of the following inputs, what does the scratch program output?

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | **0** |
| **2** | **2** |
| **3** | **3** |
| **4** | **6** |
| **5** | **6** |
| **6** | **12** |
| **7** | **12** |
| **8** | **20** |

**4.d.** What does this program do?

Sums the even integers from 2 to the input value

**4.e.** Look up the formula for the sum of the even integers from 2 to N. Write it below. Use the formula to calculate the sum for N = 6 and compare to the above table for input = 6.

Formula: SUM = M(M+1)   
How to Find M: M = ((First Even + Last Even)/2) - 1   
  
For N = 6: M = (2+6)/2 - 1 = 3

Apply the formula for SUM

SUM = 3(3+1) = 12

**Question 5.**

The computer is an incredibly powerful tool for modeling and simulation. This has uses from weather prediction to anticipating disease outbreak and spread to business modeling to physics simulation. A key component of this application is the ability to generate random events. In this exercise, you will generate random numbers, the basis of modeling random events.

**5.a.** Write the pseudocode and implement a Scratch that does the following: Ask the user for an integer greater or equal to 2. Call this variable N. Sum N random integers between 1 and 10. Print the sum and the average of the N numbers. E.g., if N = 5, you might get the following numbers: 2, 7, 9, 2, 3. The sum is 23 and the average is 23/5 or 4.6.

Random numbers can be generated in Scratch using the *pick random* block in the operators list. Be sure you understand how this works. Play with it using a *say* block. If you don’t understand, ask the instructor.

Note to grader: Students may use REPEAT UNTIL with a counter instead of REPEAT N, which is fine. Likewise, they may use REPEAT N in the pseudocode, which is fine. Also, the may get different sum and average values, but as N gets large, they should be around 5.5 as the average.

**Write “Enter an integer greater than or equal to 2”**

**INPUT N**

**SET index to N**

**SET sum to 0**

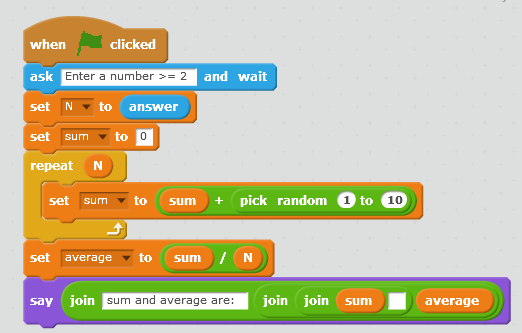
**WHILE (index > 0)**

**SET sum to sum + random\_pick\_between\_1\_and\_10**

**Set index to index – 1**

**SET average to sum/N**

**Print “The sum and the average are: ”, sum, average**



**5.b.** Fill in the following table (use 2 decimal places for the average).

|  |  |  |
| --- | --- | --- |
| **N** | **Sum** | **Average** |
| 5 | 30 | 6 |
| 10 | 62 | 6.2 |
| 20 | 134 | 6.7 |
| 50 | 282 | 5.64 |
| 100 | 535 | 5.35 |
| 1000 | 5440 | 5.44 |

**5.c.** Comment on the results. E.g., is the average what you expected? What do you think it should be?

I would expect the average to be at the half way number between 1 and 10, which is 5.5. The bigger N is, the closer you should be to this. It is the same as taking lots of samples, like in polling or rolling dice, for example. The more samples, the better the estimate. Note: In our case, each number from 1 to 10 is equally likely. This is called a uniform distribution. There are other possibilities for random processes. Think about the “bell curve” also known as the normal distribution.

**5.d** GRADER: 5.d does not count as a part of this question since this is only a submit. So this question has 3 parts, a-c.

1. Save the program using the name *Lab8\_random.sb2*.
2. Submit the program to Sakai.