15110 PRINCIPLES OF COMPUTING – LAB EXAM 2 –FALL 2012 A

Name _____ Section _____ Andrew ID _____ Machine_____

Directions:

1. *In your home directory (as soon as you start the terminal), create a folder named labexam2.*
2. *Write a function in Ruby for each of the following problems using gedit and store these functions in the labexam2 folder. Use the Ruby Reference Sheet to assist you with syntax issues.*
3. *Test your functions by calling them within irb. Although we give you sample test runs, your function should work completely based on the given specifications and your output should match the sample usage as closely as possible for full credit. Remember that we will run your code on additional test cases that are not shown on the exam.*
4. *Once you are finished, compress the labexam2 folder into a zip file and submit it to AutoLab (http://autolab.cs.cmu.edu) by the end of lab. Do not delete the labexam2 folder from your home directory.*

1. (25 pts) Write a Ruby function `f1()` (in the file `f1.rb` in your `labexam2` folder) that draws a bull's-eye target centered in a window of size 450 by 450. The target consists of concentric red and white circles whose radii are 50, 100, and 150 pixels. Note: the order in which you draw the circles matters! Start with the largest one. The result should look just like the picture below, but if you're off by a pixel or two, it's okay. Refer to the Ruby Reference Sheet for help with Canvas method syntax.

2. (25 pts) Write a function `f2(matrix)` that takes a matrix (an array of arrays) of strings as input, and returns a matrix of their lengths. Use the following algorithm:

1. Create a variable *result* as an array of the same length as the input, *matrix*.
2. For each row in *matrix* – each row will be an array of strings – compute an array containing their lengths. (You could do this with a for loop, or with the collect method.)
3. Store the array you computed in the corresponding row of *result*.
4. When you've filled in all the rows, return *result*.

Sample usage:

```
>> f2([["a","woof"], ["banana", "avocado"]])
=> [[1, 4], [6, 7]]

>> f2([["rutabaga"]])
=> [[8]]
```

3. (25 pts) In the file `f3.rb` in your `labexam2` folder, write a recursive function `f3(list)` that returns the product of all the numbers in the input `list`. Do not use a for loop; you must use recursion.

Sample usage:

```
>> f3([2, 3, 7])
=> 42

>> f3([2,5,2,5])
=> 100

>> f3([])
=> 1
```

4. (25 pts) Write a function `f4(matrix)` (in the file `f4.rb` in your `labexam2` folder) that takes as input a matrix containing a mix of integers and symbols, and returns the number of elements that are integers. You may assume that each row of the matrix has the same number of elements in it. Hint: you can use `kind_of?(Integer)` to check if something is an integer.

Sample usage:

```
>> f4([[5, 1], [:blueberry, 12]])
=> 3

>> f4([[1,2,:moose,4],[5,:squirrel,:boris,7],[:natasha,10,11,13]])
=> 8

>> f4([[:north, :west]])
=> 0
```