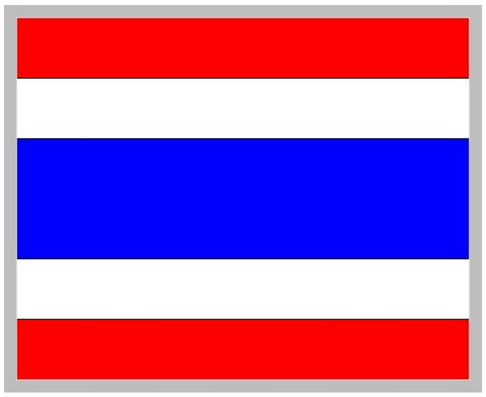
15110 PRINCIPLES OF COMPUTING – LAB EXAM 2 –FALL 2012 B

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 the flag of Thailand in a window of size 450 by 360. The flag consists of horizontal red, white, and blue bars, with the blue bar (in the center) twice as thick as the others. The top left corner of the blue bar is at x=0, y=120. 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 integers as input, and returns a matrix of bools indicating whether each integer is odd. 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*, which will be an array of integers, compute an array containing the corresponding bool value (true if the element is odd, else false.) You could do this with a for loop, or with the collect method.
   3. Store the array in the corresponding row of *result*.
   4. When you've computed all the rows, return *result*.

Sample usage:

```
>> f2([[1,2,3],[4,5,6]])
=> [[true, false, true], [false, true, false]]

>> f2([[42]])
=> [[false]]
```

3. (25 pts) In the file `f3.rb` in your `labexam2` folder, write a <u>recursive</u> function `f3(list)` that prints the elements of `list`. If an element is 100 or greater, it should be followed by a newline; otherwise it should be followed by a space. Do not use a for loop. The function should return nil.

Sample usage:

```
>> f3([2, 3, 105, 7, 9])
2 3 105
7 9 => nil

>> f3([300,400,2,5,2,500])
300
400
2 5 2 500
=> nil

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

4. (25 pts) Write a function `f4(matrix)` (in the file `f4.rb` in your `labexam2` folder) that takes as input a matrix of numbers and returns *true* if the input is an identity matrix. An identity matrix has ones along the major diagonal (where i = j) and zeros everywhere else, e.g., if M is a 3×3 identity matrix then M[0][0], M[1][1], and M[2][2] would be 1, and all the remaining elements M[i][j] where i ≠ j would be 0. Your function must work for <u>any</u> NxN matrix. Hint: go through all possible combinations of i and j, and verify that each element of *matrix* has the correct value.

Sample usage:

```
>> f4([[1,0,0],[0,1,0],[0,0,1]])
=> true

>> f4([[1,2,3],[0,1,0],[0,0,1]])
=> false

>> f4([[1]])
=> true
```