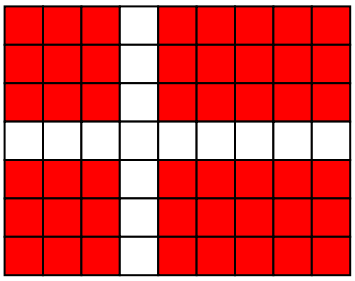
15110 PRINCIPLES OF COMPUTING – LAB EXAM 2 – SPRING 2012 A

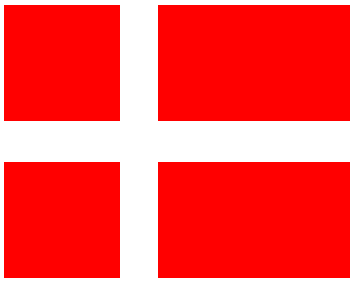
Name _____ Section _____ Andrew ID _____ Machine_____

Directions:

1. *In your home directory, 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 displays the flag of Denmark in a window of size 450 by 350. The flag consists of a red background with two white stripes, each with a width of 50 pixels. The grid is shown as a reference guide for you to estimate the proper size of the elements. DO NOT DRAW THE GRID!





2. (25 pts) Write a function `f2(n)` (in the file `f2.rb` in your `labexam2` folder) that computes 2^n recursively for a non-negative integer n, using the following definition: if n > 0, then $2^n = 2(2^{n-1})$. Your solution must use recursion and should NOT use iteration. Your function should not use the exponentiation (**) operator.

Sample usage:

```
>> f2(4)          >>f2(1)          >> f2(7)          >> f2(0)
=> 16             => 2             => 128            => 1
```

(OVER)

3. (25 pts) In a new forest fire simulation, you need a matrix of integers representing trees. The value of each integer represents the tree's current state: 2 = on fire, 3 = in ashes, 4 = sprout, 5 = small tree, 6 = medium tree, 7 = mature tree. Write a function `f3(numrows,numcols)` (in the file `f3.rb` in your `labexam2` folder) that takes two integers representing the size of the matrix in rows and columns. Your function should return a matrix of the given size with each integer in the matrix assigned to a random integer between 2 and 7 (inclusive). Use the rand function to generate random integers. You may assume that numrows and numcols are integers and are both greater than 0.

Use the following algorithm:

1. Define *matrix* to be a new one-dimensional array of size *numrows*.
2. For each element in *matrix*, do the following:
   a. Initialize the current element in *matrix* to a new one-dimensional array of size *numcols*.
3. For each row of *matrix* do the following:
   a. For each column of *matrix* do the following:
      i. Initialize the current row and column of *matrix* to a random integer in the given range.

Sample usage:

```
>> srand(15110); nil
=> nil
>> f3(5,4)
=> [[3,2,2,3], [5,6,4,6], [6,2,5,4], [4,4,6,2], [2,6,3,6]]
```

4. (25 pts) Write a function `f4(matrix)` (in the file `f4.rb` in your `labexam2` folder) that takes a matrix of integers and returns the total number of integers that are odd. You may assume that each row of the matrix has the same number of integers in it. You may also assume that all integers in the matrix are positive.

Sample usage:

```
>> f4([[1,2,3,4],[5,7,7,7],[9,10,11,13]])
=> 9

>> f4([[4,6,8],[2,8,12],[4,20,18],[12,42,28],[26,36,70]])
=> 0

>> srand(15110); nil
=> nil
>> f4(f3(5,4))
=> 5
```

(the last example is based on the sample run from problem 3)