CSCI 136 Written Exam #2 **Name: ___________________________________**
Fundamentals of Computer Science II
Spring 2015

This exam consists of 6 problems on the following 6 pages.

You may use your double-sided hand-written 8 ½ x 11 note sheet during the exam. No computers, mobile devices, cell phones, or other communication devices of any kind are permitted.

If you have a question, raise your hand and I will stop by. Since partial credit is possible, **please write legibly and show your work**.

Problem	**Points**	**Score**
1	**10**	
2	**10**	
3	**10**	
4	**10**	
5	**10**	
6	**10**	
Total	**60**	

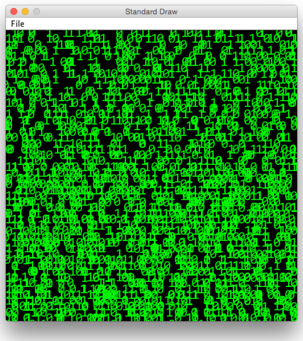
1. Input, output, classes (10 points). Consider the following program:

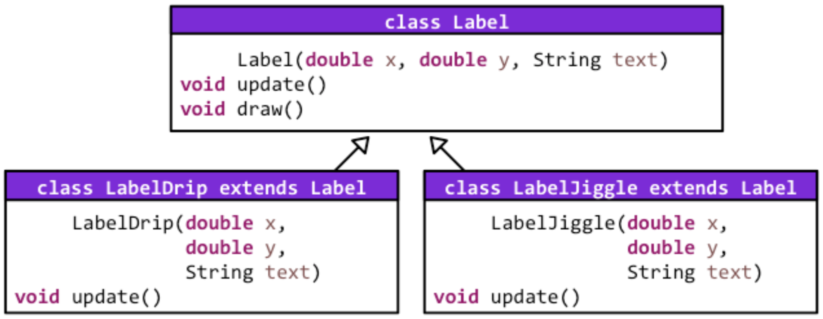
```
public class Prob1
{
    private boolean big = false;
    public Prob1(String str)
    {
        big = (str.length() > 3);
    }

    public String toString()
    {
        if (big)
            return "big! ";
        return "small! ";
    }

    public static void main(String[] args)
    {
        Prob1 a = new Prob1(args[0]);
        Prob1 b = new Prob1(args[1]);
        System.out.printf("%d: %s %s\n", args.length, a, b);
    }
}
```

Below are five example executions of the program. Give the output produced by the program. If the given input would cause a runtime error, write "error".

Command line	Output
`% java Prob1 the`	
`% java Prob1 the cat`	
`% java Prob1 the cat sat`	
`% java Prob1 a frog`	
`% java Prob1 -485 12`	

2. Multidimensional arrays, inheritance (10 points). Fill in the missing code in the following program that animates 0's and 1's that stay still, drip down, or jiggle by making use of the `Label`, `LabelDrip`, and `LabelJiggle` classes. Here is an example run along with the class hierarchy and APIs:





```
class Label
      Label(double x, double y, String text)
void update()
void draw()

class LabelDrip extends Label
      LabelDrip(double x,
                double y,
                String text)
void update()

class LabelJiggle extends Label
      LabelJiggle(double x,
                  double y,
                  String text)
void update()
```

```
public static void main(String[] args)
{
   final int COLS = Integer.parseInt(args[0]); // First arg is rows of 0/1's
   final int ROWS = Integer.parseInt(args[0]); // Second arg is columns of 0/1's

   ______________ [][] labels = __________________________________; // Create 2D array

   for (int col = 0; col < COLS; col++)
   {
      for (int row = 0; row < ROWS; row++)
      {
         double x = 1.0 / COLS * (col + 0.5);
         double y = 1.0 / ROWS * (row + 0.5);
         String num = "0";

         if (___________________________________) // 50% chance of changing num to "1"
            num = "1";

         Label label = null; // Randomly create one of our three label data types
         double rand = Math.random();
         if      (rand < 0.333) label = _________________________________; // Normal

         else if (rand < 0.666) label = _________________________________; // Dripping

         else                   label = _________________________________; // Jiggling

         ___________________________________________; // Put this label into 2D array
      }
   }

   ______________________________________  // Loop forever
   {
      // Omitted: clearing drawing window, setting pen color, sleep, etc.
      for (int col = 0; col < COLS; col++)
      {
         for (int row = 0; row < ROWS; row++)
         {
            _______________________________________  // Ask labels to update positions

            _______________________________________  // Ask labels to draw themselves
         }
      }
   }
}
```

3. Regular expressions (10 points). Match the description of a set of strings on the left with the regular expression on the right that matches exactly this set of strings. For the purposes of this problem, assume the following definitions:

- *Binary number* – consists of only the numbers `0` and `1`, e.g. `101010`
- *Hexadecimal number* – consists of only the numbers `0` through `9` plus the letters A through F (uppercase), e.g. `05F6A`

Also you can assume the empty string is ***not*** considered a binary or hexadecimal number.
Letters will be used exactly once.

____ Hexadecimal numbers with exactly two digits

____ Binary numbers with an even number of digits

____ Binary numbers with an odd number of digits

____ Binary numbers in which the sequence 010 appears at least once

____ Hexadecimal numbers that end with F and are more than three digits long

____ Hexadecimal numbers that end with F and are at most three digits long

____ Binary numbers that have at least two zeros

____ Binary numbers that have at least a single one

____ Hexadecimal numbers that start and end with F

____ Hexadecimal numbers that do not contain the digits A-F

A. `[0-9A-F]{3,}F`

B. `([01][01])+`

C. `[0-9A-F][0-9A-F]`

D. `[01]*0[01]*0[01]*`

E. `[0-9A-F]{0,2}F`

F. `[01]([01][01])*`

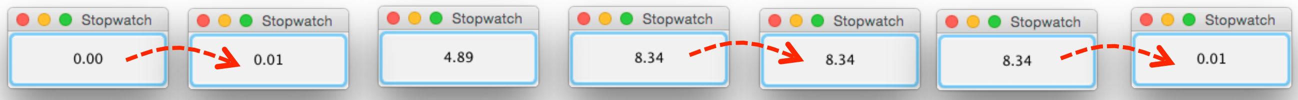
G. `[01]*010[01]*`

H. `[0-9]+`

I. `[01]*1[01]*`

J. `F | F[0-9A-F]*F`

4. GUIs, threading (10 points). You are building a simple stopwatch GUI. The GUI has a single button that starts labeled 0.00. When the user click the button, it starts displaying the elapsed time in seconds to two decimal places. If the button is clicked again, the stopwatch halts and is left displaying the elapsed time since it was started. Clicking it again causes the timer to be reset and start counting up. Below is the GUI in action. Transitions with an arrow denote clicking the button. Transitions without an arrow are time passing.



Put letters into the underlined spaces to create the described GUI. ***Letters may be used 0 or more times.***

```java
public class Stopwatch _____ _____
{
  private JButton button = _____;
  private Stats stats = new Stats();
  private boolean running = false;

  public Stopwatch()
  {
    JFrame frame = new JFrame("Stopwatch");
    frame.getContentPane().add(button);

    button._____;

    Thread t = _____;
    t.start();

    frame._____(150, 75);
    frame.setVisible(true);
  }

  public void _____
  {
    while (true)
    {
      if (running)
      {
        String t = String.format(_____, stats.elapsedTime());
        button.setText(t);
      }
      try { Thread.sleep(10); }
      catch (InterruptedException e) {}
    }
  }

  public void ______
  {
    running = _____;
    if (running)
      stats.reset();
  }

  public static void main(String[] args)
  {
    Stopwatch watch = new Stopwatch();
  }
}
```

```
A. extends
B. implements
C. false
D. true
E. !running
F. Runnable, ActionListener
G. Comparable
H. new JButton("0.00");
I. setEnabled
J. setSize
K. setText
L. add(button);
M. reset(button);
N. addMouseListener(this);
O. addActionListener(this);
P. addActionListener(button);
Q. actionPerformed(ActionEvent e)
R. mousePressed(MouseEvent e)
S. run()
T. new Thread();
U. new Thread(this);
V. "%2d"
W. "%e"
X. "%.2f"
Y. JFrame
Z. JPanel
```

5. Algorithms, conditionals (10 points). Fill in the missing code so each method behaves as described.

```
// Returns the minimum value in the array vals.
// Returns Double.NaN if vals has a length of zero.
public static double min(double [] vals)
{
   if (_______________________________________________________________________)
      return Double.NaN;

   double result = vals[0];
   for (int i = 1; i < vals.length; i++)
   {
      if (_______________________________________________________________________)
         result = vals[i];
   }
   return result;
}

// Returns true if the vals array contains at least one even number.
// Returns false if all numbers are odd, or if vals has a length of zero.
public static boolean containsEvenNumber(int [] vals)
{
   for (int i : vals)
   {
     if (_______________________________________________________________________)
        return true;
   }
   return false;
}

// Returns true if all elements in vals are in interval [low, high] (inclusive).
// Returns false if any number in vals is not in [low, high].
public static boolean allInRange(int [] vals, int low, int high)
{
   for (int i : vals)
   {
      if (_______________________________________________________________________)
         return false;
   }
   return true;
}

// Returns true if vals1 and vals2 have at least one value in common.
// Returns false if no number is shared, or if vals1 and/or vals2 contains no values.
public static boolean intersects(int [] vals1, int [] vals2)
{
   for (int i : vals1)
   {
      for (int j : vals2)
      {
         if (_______________________________________________________________________)
            return true;
      }
   }
   return false;
}
```

6. Classes (10 points). Assume you have the following hierarchy of Java classes (some details are omitted):

```
public abstract class Item
{
   private String name = "";

   public String getName()     { return name; }

   public abstract String toString();
}

public abstract class Product extends Item
{
   private double cost = 0.0;

   public double getCost()     { return cost;       }
   public boolean isFree()     { return cost > 0.0; }
}

public class Available extends Product
{
   private int quantity = 0;

   public int  getQuantity()   { return quantity;    }
   public void buy(int amount) { quantity -= amount; }
   public String toString()    { return String.format("%d %s %.2f",
                                         quantity, getName(), getCost()); }
}

public class Discontinued extends Product { /* stuff */ }

public class Recalled extends Product     { /* stuff */ }
```

a) What method(s) must be implemented in `Discontinued` and `Recalled` in order for them to compile?

b) Circle the concrete class(es) you could successfully instantiate an object of:

```
Item     Product     Available     Discontinued     Recalled
```

c) Circle the concrete class(es) you could instantiate and add to the following variable named `list`:

```
ArrayList<Product> list = new ArrayList<Product>();

Item     Product     Available     Discontinued     Recalled
```

d) Given `list` from part c, provide a code snippet that prints out a string representation of each item.

e) Circle the inherited method(s) that you could *override* in the class `Recalled`.

```
int getQuantity()      void buy(int amount)     double getCost()

boolean isFree()       String getName()
```