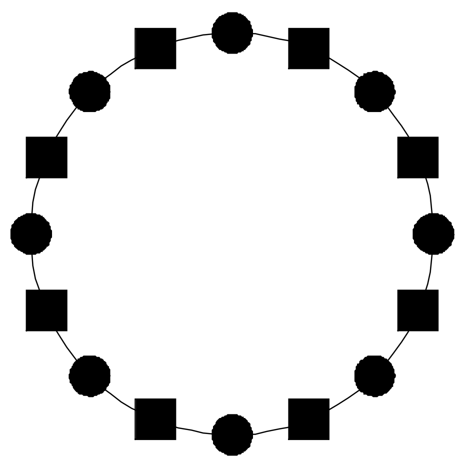
CS 100: Section Assignment 2

(For the week of February 8)

Section assignments are discussed in section and are not submitted for grading. They relate to recent lecture topics and usually to the current Programming Assignment. Prelim questions are based on Section Assignments, Programming Assignments, and Lecture examples.

1. Let $s = 1 + \sqrt{3}$. Write a program that prints a 15-line table. On the *n*th line should appear *n*, the next integer bigger than s^{2n}, 2^{n+1} and the remainder when this next largest integer is divided by 2^{n+1}. Use type `long` and make full use of `Math.pow.` Use `Math.ceil` to get the required next largest integer. After you figure that out, rewrite your program so that it prints exactly the same table but does not use the `Math.pow` method.

2. Write a program that produces the following graphic:



Think of this as a circle with trinkets. The circle should be centered at (h_c, v_c) = (400,300) and should have radius r_1 = 200. There should be n = 16 trinkets. For k = 0,…, n-1, the kth trinket has center $(h_c + r_1\cos(k\theta), v_c - r_1\sin(k\theta))$ where $\theta = 2\pi/n$. With this scheme we index trinkets counterclockwise identifying the “3 o’clock” trinket as trinket 0. The *k*th trinket is a square with side $2r_2$ if k is odd and a circle with radius r_2 if k is even. Set r_2 = 20. Use `fillOval` and `fillRect` for drawing trinkets. They work just like `drawOval` and `drawRect` only they fill in the shape. Pay attention to types. Encapsulate the “design parameters” as constants.

Hints: Use a for-loop that counts from 0 to n-1 to to draw the trinkets. Each time through the loop you should draw either a square or a circle. The remainder of the loop index divided by 2 can be used to figure out which.

3. The Fibonacci sequence is given by $f_{-1} = 0$, $f_0 = 1$, $f_1 = 1$, $f_2 = 2$, $f_3 = 3$, $f_4 = 5$, $f_5 = 8$, $f_6 = 13$, $f_7 = 21$, etc. In general, the *n*th Fibonacci number is the sum of its two predecessors, i.e., $f_n = f_{n-1} + f_{n-2}$. Here is a program that prints $f_1, \ldots, f_{20}$:

```
// S2_3
import java.io.*;

public class S2_3
{
   public static void main(String args[])
   {
      final int kmax=20;  // index of the last Fibonacci number to be printed
      int   k;            // index of the current Fibonacci number
      long z;             // the current Fibonacci number
      long y;             // the "parent" of z.
      long x ;            // the "grandparent" of z.

      TokenReader in = new TokenReader(System.in);

      x = 0;
      y = 1;
      z = 1;
      for(k=1;k<=kmax;k=k+1)
      {
         System.out.println("  " + k + "   " + z);
         x = y;
         y = z;
         z = x + y;
      }
      in.waitUntilEnter();
   }
}
```

Execute a few passes through the loop by hand so that you understand the roles of the variables `x`, `y`, and `z`. (a) Convert the `for`-loop to a `while`-loop. The modified program should print exactly the same thing. (b) Modify the part (a) program so that instead of printing f_1 ,…, f_{20} it prints all Fibonacci numbers that are strictly less than one billion. (The program need not print f_{-1} and f_0 .)

4. Here are two definitions:

- An integer is a “type A integer” if it does not have 2, 3, 5, or 7 as a divisor.
- An integer is a “type B integer” if its ones place value, tens place value, and hundreds place value are not distinct.

Note that 66 is not a type A integer because it is divisible by 3. On the other hand, 143 = 11*13 is a type A integer. Note that 234 is not a type B integer because its ones place value is 4, its tens place value is 3, and its hundreds place value is 2 and these three digits are distinct. On the other hand, 707, 991, 344, and 555 are all type B integers.

Write a single program that prints the number of type A integers that are less than or equal to one million and the number of type B integers that are from the set {100,101,…,999}.