COMP-202A, Fall 2007, All Sections
Assignment 2

Due: Tuesday October 9, 2007 (23:55)

You **MUST** do this assignment individually and, unless otherwise specified, you **MUST** follow all the general instructions and regulations for assignments.

Note that for this and all further assignments, graders **will generously** deduct marks for assignments that do not follow instructions and regulations.

Failure to respect instructions and regulations:	0 to -10 points
Part 1:	0 points
Part 2, Question 1:	6 points
Part 2, Question 2:	4 points
	10 points total

Part 1 (0 points)

Do not hand in this part. It will not be graded. But doing this exercise might help you to do the second part of the assignment (that will be graded). If you have difficulties with the questions of Part 2, then we suggest you go to one of the office hours. The TA can help you and work with you through the warm-up questions.

Warm-up Question 1 (0 points)

The following is a program determines if an input number is prime. There are two loops; try changing the first loop into a *while* loop. The second loop keeps iterating even after it has found the number to be not prime; can you make it more efficient?

```
/*********************************************************
A program that determines whether a given number is prime
*********************************************************/
import java.util.Scanner;

public class PrimeCheck {

  public static void main(String[] args) {
```

```
    long count,number,maxcheck;
    Scanner scan = new Scanner(System.in);

    do {
        System.out.print("Enter the number to check: ");
        number = scan.nextLong();
        if (number<2)
            System.out.println("It must be greater than 1.");
    } while(number<2);

    // to see if it's prime we check to see if it's divisible by any
    // number between 2 and its square root
    maxcheck = (long)Math.sqrt(number);

    // assume it is prime unless we prove otherwise
    boolean isPrime = true;
    count = 2;
    while (count<=maxcheck) {
      if (number%count==0) {
        // divides evenly: not prime
        isPrime = false;
      }
      count++;
    }
    System.out.println("Your number "+number+
                       ((isPrime)? " is" : " is not")+" prime.");
  }
}
```

Part 2 (6 + 4 = 10 points)

The questions in this part of the assignment will be graded.

Question 1 (6 points)

The *Collatz Conjecture* is a problem posed many years ago, well-known for consuming a great deal of time by researchers (unsuccessfully) trying to prove or disprove it. Given an integer $n \geq 1$, we compute the function $f(x)$ as follows:

$$f(x) = \begin{cases} 3 \times x + 1 & \text{if } x \text{ is odd,} \\ x/2 & \text{if } x \text{ is even} \end{cases}$$

For example, the number 10 is even so $f(10) = 5$. If we apply f to the output result we get $f(5) = 16$ since 5 is odd. We can keep applying f to the result of the previous

application: $f(16) = 8$, $f(8) = 4$, $f(4) = 2$, $f(2) = 1$, $f(1) = 4$. At this point we find we are in a cycle, $f(1) = 4, f(4) = 2, f(2) = 1$. The *Collatz conjecture* states that for all $n \geq 1$ we eventually reach this cycle if we keep iteratively applying f to own output.

Your task is to verify the *Collatz conjecture* for all integers from 1 to 2^{16}. Write a program called "Collatz.java" that implements f and then loops through all the indicated numbers. For each number n it should compute $f(n)$, then f of the result etc, until it finds it has reached the 1,4,2 cycle (eg reached 1). At the same time keep track of how many iterations it takes a given number to reach the cycle. Your program should output the maximum number of iterations to reach the cycle taken by any number in the range 1 to 2^{16} as well as the number that generated that maximum.

A suggested approach:

- Write code that computes $f(x)$ and returns the result.
- Write code that executes $f(x)$, then executes f on the result again (ie $f(f(x))$), etc., and which checks for the end of the cycle to know when to stop.
- Once you're sure that code works add code to keep count of the number of the number of iterations required to reach the cycle.
- Write code that does the above for all the required numbers and keeps track of the maximum number of iterations.

Question 2 (4 points)

A very simple game is played by two players. 5 rows of sticks are arranged with the first row consisting of 1 stick, 2nd row consisting of 2 sticks and so on. Players take turns; they can remove any number of sticks they want, but they must all be from the same row. The player who wins is the one who takes the last stick.

Construct a program called "StickGame.java" that allows two people to play this game. Your game should alternate between two players, first letting player A make a move, and then letting player B, etc. Before each move the board should be printed out. After each move the game should be checked to see if the player has won. If not the players switch turns and play continues.

You will need code that does the following:

- So the players can see how many sticks are left in each row you should print out an easily understood visual representation, not just numeric counts of sticks and rows. E.g., use vertical bars ("|") for each stick left in the game and use 5 lines to show the entire board.
- Ask for how many sticks and from which row the user wants to remove sticks, then ask how many, and then do so. You do not need to check that the responses are valid.
- Determine whether the player has won or not.

When the game is over you should emit a message indicating who won.