CS100M Spring 2007: Project 4 Grading Guide

The coded items below (e.g., c1e, s2a) indicate what a student's solution should accomplish. Codes that begin with the letter 'c' deal with correctness; codes that begin with 's' deal with style.

Grader: If a student's solution does not accomplish task c1a, for example, then write the task code 'c1a' along with any diagnostic remarks you can give. Count the number of correctness and style errors separately.

Items marked with ** count as two errors. In the table below, the top row lists the possible scores (1 to 5). The next row lists the number of correctness errors corresponding to every score category. The style score is determined similarly. Enter the total score (maximum of 10) in CMS as the project score. If there are bonus questions, enter any bonus points separately in the "Bonus Bucket," separate from the project score.

Student: Read the grading guide for every project, even if you get a perfect score! Notice from the table below that we often give one or two "freebies," i.e., mistakes that don't cost you any points. Learn from working on the project, and learn from any mistakes.

Scores

- c and s stand for correctness and style; see table below.
- parts with ** next to them means that they are double the value, *** for triple, etc.
- Apply bonus for exemplary work or doing additional tasks.

Score	0	1	2	3	4	5
#correctness errors	>9	7-9	5,6	3,4	2	0,1
# style errors	>10	8-10	5-7	3,4	2	0,1

General

(s0a) Use meaningful variable names
(s0b) Appropriate indentation
(s0d) Appropriate and concise comments throughout
(s0e) Reasonable line lengths; no horizontal scrolling
(s0f) [up to **] No superfluous code
(s0g) **[up to ***] Each class has a class comment header**

(c0a) [up to **] Program compiles without error. (1 * for each compiler error message up to 2)
(c0b) [up to **] Program successfully executes without crashing. (* for occasional, ** for persistent)
(c0c) [up to **] Class name matches filename.
(c0d) ** Do not use arrays in this project, as described in the project specification.

1. Calculator

```
a: 1628
b: 2
c: 2.176470588235294
d: 5.150000000000001
e: 2
f: -6
g: 11
h: true
i: true
j: false
k: 1048576.0
l: 1.0
m: 10
n: a
```

(c1a) Output values and types are correct. For (k), integer and real output are both acceptable.

(c1b) Don't use Math.pow in (j).

(c1c) In (m) and (n), random number distribution should be correct.

(s1a) Program prints out labels.

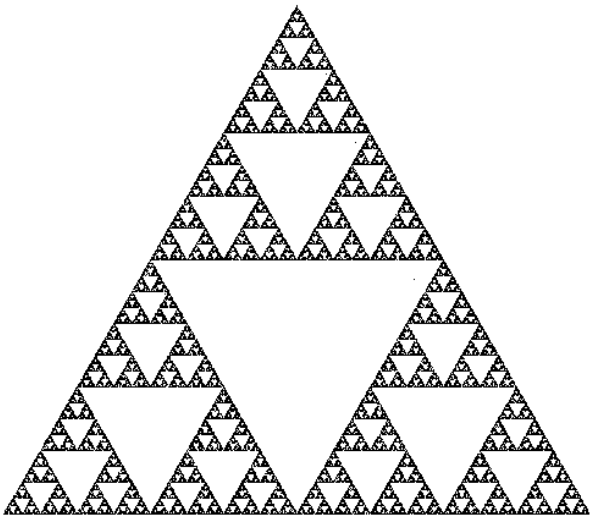
For correctness, each question counts only once.
(s1a) counts only once.

Final correctness error # =

#Questions wrong	1-3	4-9	10-12	>12
#correctness errors	1	2	3	4

2. Chaos Game

The final image should be like the following.



(c2a) Distribution of the random numbers is correct.
(c2b) Correctly use the random numbers in conditionals to determine next point to draw.
(c2c) Correctly calculate the next point
(c2d) Correctly update the variable for the points
(c2e) Correctly control the loop
(c2f) Correctly use MyFrame class to draw the points

3. Barnsley's Fern

The final image should be like the following. If not, check if the random number is of the correct distribution. If the image is too thin, check if the coordinates are updated simultaneously.

(c3a) *biasedFour* is defined as a static function that resturns an integer and is used in the *main* method.
(c3b) *biasedFour* correctly generates the required distribution.
(c3c) Display range is correctly set with `f.setRange` so that image can be viewed in full.
(c3d) Correctly calculates the fern functions.
(c3e) Simultaneously update the coordinates using temporary variables or other ways.
(c3f) Correctly use if statements to apply the fern functions.
(c3g) Correctly use MyFrame class to draw the points

(s3a) Concise method header comment given above the *biasedFour* method

```java
////////////////////////////////////////////////////////////////////////////////
// Calculator.java
////////////////////////////////////////////////////////////////////////////////

public class Calculator {

 public static void main(String[] args) {

  System.out.println("a: " + (117+31)*11);
  System.out.println("b: " + 37/17);
  System.out.println("c: " + 37.0/17);

  // The remainder of 74.23/6.28
  System.out.println("d: " + 74.23%6.28);

  // The quotient of 34/13
  System.out.println("e: " + 34/13);

  // The remainder of -87/9
  System.out.println("f: " + -87%9);

  // Find the quotient (integer) for 74.23/6.28. Hint: use casting.
  System.out.println("g: " + (int)(74.23/6.28));

  // The result of evaluating 23 is greater than 11 and less than 45.
  System.out.println("h: " + (23>11 && 23<45));

  // The result of evaluating 2^2 is not equal to 5.
  System.out.println("i: " + (2*2 != 5));

  // Evaluate the expression sqrt(37.001))^2 is equal to 37.001
  // (To do this you may use Math.sqrt but not Math.pow)
  System.out.println("j: " + ((Math.sqrt(37.001)*Math.sqrt(37.001)) == 37.001));

  // The bigger value between 2^20 and 3^12
  System.out.println("k: " + Math.max(Math.pow(2,20), Math.pow(3,12)));

  // Evaluate the expression sin(PI/4)^2 + cos(PI/4)^2
  System.out.println("l: " + (Math.sin(Math.PI/4) * Math.sin(Math.PI/4)
        + Math.cos(Math.PI/4) * Math.cos(Math.PI/4)));

  // Generate a random integer on the interval [-1, 12]
  System.out.println("m: " + (int) Math.floor((Math.random()*13) - 1));

  // Randomly print one letter from the set . The
  // probability of each letter should be equal.) (Use Math.random)
  System.out.println("n: " + (char)((int) Math.floor(Math.random()*5)+'a'));
 }
}



////////////////////////////////////////////////////////////////////////////////
// Chaos.java
////////////////////////////////////////////////////////////////////////////////

import java.awt.Color;

/* Generate the Sierpinski triangle
 */
public class Chaos {

  public static void main(String[] args) {

    MyFrame f = new MyFrame("Chaos Game");

    double x = 0.4, y = 0.8;
    int k = (int) Math.pow(3, 9);

    for (int i=0; i<k; i++) {

      int n = (int) Math.floor(3 * Math.random()+1);
      if (n==1) {
        x = 0.5 * (x + 0);
        y = 0.5 * (y + 0);
      } else if (n==2){
        x = 0.5 * (x + 1);
        y = 0.5 * (y + 0);
      } else {
        x = 0.5 * (x + 0.5);
```

```
      y = 0.5 * (y + 0.866);
    }

    f.drawPoint(x, y, Color.black);
  }
 }
}


/* Fern.java
 */

import java.awt.Color;

/* Generate Barnsley's Fern
 */
public class Fern {

  public static void main(String[] args) {

    MyFrame f = new MyFrame("Fern");
    f.setRange(-3, 3, -1, 11);

    double x = 0.4, y = 0.8;
    double newx, newy;

    for (int i=0; i<200000; i++) {

      int n = biasedFour();

      if (n == 1) {
        newx = 0.85*x + 0.04*y;
        newy = -0.04*x + 0.85*y + 1.6;
        x = newx; y = newy;
      } else if (n == 2) {
        newx = -0.15*x + 0.28*y;
        newy = 0.26*x + 0.24*y + 0.44;
        x = newx; y = newy;
      } else if (n == 3) {
        newx = 0.2*x - 0.26*y;
        newy = 0.23*x + 0.22*y + 1.6;
        x = newx; y = newy;
      } else if (n == 4) {
        newx = 0;
        newy = 0.16*y;
        x = newx; y = newy;
      }

      f.drawPoint(x, y, Color.black);
    }
  }

  /* Returns 1 with probability (p) = 0.85,
   *         2 with p=0.07,
   *         3 with p=0.07,
   *         4 with p=0.01
   */
  public static int biasedFour()
  {
    double r = Math.random();
    if (r<=0.85) return 1;
    else if (r<=0.92) return 2;
    else if (r<=0.99) return 3;
    else return 4;
  }
}
```