CS241, Homework 1 – Java and OOP fundamentals, Big-Oh

1. Suppose we create a Coin class that is designed to represent a coin that can be tossed to come up with heads or tails, each with a 50% chance. Each coin object should also store a record of all the tosses it has made.

```
public class Coin {
  ______ ArrayList<Boolean> pastTosses;  // stores a record of each toss:
                                         // true=heads, false=tails
  public Coin() {
    pastTosses = new ArrayList<>();
  }

  public ______ toss() {
    if (Math.random() >= 0.5) {
      // do something
    }
    else {
      // do something else
    }
    // maybe more code?
  }

  public boolean getLastToss() { // you will add code here
  }
}
```

   a. Should the pastTosses instance variable be public, private, or protected? Explain your choice.

   b. Should toss() have a return value? Explain your answer.

   c. Explain how you would fill in the two (or perhaps three) blank sections marked with comments in the toss() method.

   d. Should there be a getter for pastTosses? Why or why not?

   e. Should there be a setter for pastTosses? Why or why not?

   f. Fill in the getLastToss() method.

2. Design and write code for a TrickCoin class that allows the user to specify the likelihood that the coin will come up heads vs tails. TrickCoin should be a subclass of Coin (TrickCoin extends Coin), and should have a constructor TrickCoin(double percent) that allows the user to specify the percentage of the time the coin comes up heads. percent will be a number between 0 and 1, with 0=0%, and 1=100%.

   Make sure to override toss() correctly, and add/override any other methods you deem appropriate. You can also choose to make any changes you want to the original Coin class.

   This question is intended to be done on paper, not at a computer. That is, I've purposefully made this simple enough that you should try to do it without running it in an IDE. You will not be graded on syntax. (This is the kind of question that would be appropriate for a test --- a small coding question to be done on paper).

3. **Big-oh ranking**

   Order the following big-oh complexities in order from slowest-growing to fastest-growing (this is not the same as slowest to fastest running time!) It is possible some of them are actually in the same big-oh category. If that is the case, make it clear which ones have the same complexity.

   n^2, 3^n, $\sqrt{n}$, 1, $n*\log(n)$, 2^n, $n!$, $2^{\log(n)}$, n^3, n, $n^2\log(n)$, $\log(n)$, 2^{n+1}

4. **Big-oh complexity**

   Assume each formula below represents the running time T(n) of some algorithm. For each formula, give the lowest big-oh complexity possible (the tightest bound).

   Here, log represents the base-2 logarithm.

   (a) $5 + n^2 + 25n$
   (b) $50n + 10n^{1.5} + 5n*\log(n)$
   (c) $3n + 5n^{1.5} + 2n^{1.75}$
   (d) $n^2\log(n) + n*\log(n) + n*(\log(n))^2$
   (e) $2^n + n^{10}$
   (f) $n*\log(n) + 8n + n*(\log(n))^2$

5. **Big-oh complexity proof**

   Assume we have analyzed an algorithm, and its run time is determined to be

   $T(n) = n^3 + 2n + 3$

   (a) What is the big-oh running time of this algorithm? (This should be easy.) Call this function $f(n)$.

   (b) Now, prove your answer.
   In other words, find constants $c > 0$ and $n_0 > 0$ such that for all numbers $n >= n_0$, $T(n) <= c * f(n)$. You may prove that your c and n_0 work by drawing a graph showing T(n) and $c * f(n)$. Label your axes and where n_0 is. Make sure the graph is easy to read and interpret.

6. **Big-oh complexity analysis of code**

Determine the big-oh running time for the following algorithms in terms of n. (No justification needed.)

a. Matrix addition:

```
for (int i = 0; i < n; i++)
{
  for (int j = 0; j < n; j++)
  {
    c[i][j] = a[i][j] + b[i][j];
  }
}
```

b. Matrix multiplication:

```
for (int i = 0; i < n; i++)
{
  for (int j = 0; j < n; j++)
  {
    c[i][j] = 0;
    for (int k = 0; k < n; k++)
    {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

c.

```
int counter = 0;
while (n >= 1)
{
  n = n/2;
  counter++;
}
```

d.

```
int counter = 0;
int x = 1;
for (int i = 0; i < n; i++)
{
  for (int j = 0; j < x; j++)
  {
    counter++;
  }
  x = x * 2;
}
```