Computer Science 245
Practice Test

1. Given the following method which finds the product of the elements in an array

```
static int findProduct(int[] array) {

       int product = 1;

       for (int i = 0; i < array && product != 0; i++)

            product *= array[i];

       return product;
}
```

Analyze the running time of this method, finding a big-oh category for both the best case and the worst case.

2. To minimize round-off error when summing a set of doubles, it is important to add from smallest to largest. Suppose you knew that the data you were receiving increased monotonically for a known period, and then dropped again. For example, the sequence 3 18 31 46 62 1 22 38 55 59 4 12 21 39 48 6 15 increases with a period of 5. To sum a sequence like this, you should iterate over the array in a skip-like manner, as in this method:

```
static double findSum(double[] array, int period) {

       double sum = 0;

       for (int i = 0; i < period; i++)

            for (int j = i; j < array.length; j += period)

                    sum += array[j];

      return sum;
}
```

Analyze the running time of this method, finding a big-oh category for both the best case and the worst case.

3. Consider the following code.

```
public class Matrix {
    private static Random randy = new Random();
    private int[][] internal;
    public Matrix(int n, int m, int max) {
         internal = int[n][m];
         for (int i = 0; i < n; i++)
           for (int j = 0; j < m; j++)
               internal[i][j] = randy.nextInt(max);
    }
}
```

a. What kind of variable is `randy`?

b. What kind of variable is `internal`?

c. What kind of variable is `max`?

d. What kind of variable is `i`?

4. What does it mean for ...

a. ... a variable to be final?

b. ... a class to be final?

c. ... a method to be final?

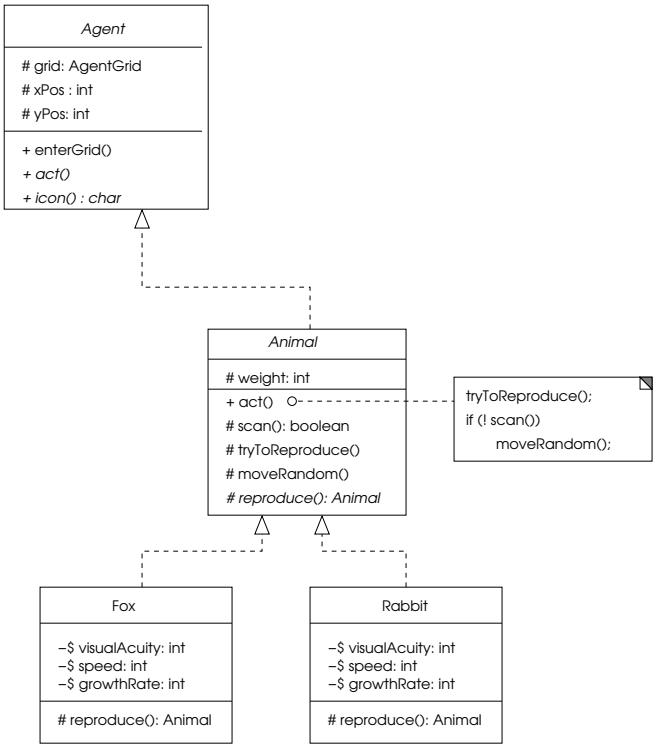
5. Based on the code on the last page, indicate which method will be called in each of the invocations below.

```
Item a = new Food();
Tent t = new Tent();
Food f = new Food();
Key k = new Key();

Dungeon d = new Dungeon();
Location y = new Yard();
```

```
d.drop(a);

d.drop(t);

d.drop(f);

d.drop(k);

y.drop(a);

y.drop(t);
```

```
y.drop(f);

y.drop(k);

t.drop(a);

t.drop(t);

t.drop(f);

t.drop(k);
```

6. Recall this portion of the design from the predator/prey example:



Suppose some animal species are are "herd" animals; a set of individuals from this species exist together as a herd, and they cannot move to a space unless it is within three spaces of another member of the herd (not just any other member of the same species; there might be more than one herd of a given species, and an individual must stay by its herd). New individuals become members of their parent's herd.

Modify this design so that new species classes that are herd classes can share code (for example, `Buffalo` and `Antelope` might both be herd classes, so they should share code relevant to herd information or behavior). Give *very* rough pseudo-code for relevant methods.

```
interface Item {}
class Food implements Item {}
class Key implements Item {}

interface Location {

    void drop(Item x);
    void drop(Food x);
}

abstract class InsideLocation implements Location {

    public void drop(Item x) { System.out.println("1"); }
    abstract void drop(Key x);
}

class Dungeon extends InsideLocation {

    public void drop(Key x) { System.out.println("2"); };
    public void drop(Food x) { System.out.println("3"); }
    public void drop(Tent x) { System.out.println("4"); }
}

class Tent extends InsideLocation implements Item {
    public void drop(Key x) { System.out.println("5"); }
    public void drop(Item x) { System.out.println("6"); }
    public void drop(Food x) { System.out.println("7"); }
}

class Yard implements Location {
    public void drop(Item x) { System.out.println("8"); }
    public void drop(Food x) { System.out.println("9"); }
}

public class Method {
    public static void main(String[] args) {

        Item a = new Food();
        Tent t = new Tent();
        Food f = new Food();
        Key k = new Key();

        Dungeon d = new Dungeon();
        Location y = new Yard();

        d.drop(a);
        d.drop(t);
        d.drop(f);
        d.drop(k);
        y.drop(a);
        y.drop(t);
        y.drop(f);
        y.drop(k);
        t.drop(a);
        t.drop(t);
        t.drop(f);
        t.drop(k);
    }
}
```