Recall the ArrayList class that we created in lecture on Monday had three fields: size, capacity, and data. The first two were `int`s and the last was an array of Strings. Implement the following constructor which takes in an initial size, size and sets each element equal to the initialValue passed to the constructor.

```
public ArrayList(int size, String initialValue) {



}
```

(a) Give the worst-case time complexity for the following method. Explain your reasoning.

```
public double stddev(double[] data) {
  if(data.length<1) {
    throw new Exception();
  }
  double sum = 0;
  for(int i=0; i<data.length; ++i) {
    sum += data[i];
  }
  double average = sum / data.length;
  sum = 0;
  for(int i=0; i<data.length; ++i) {
    sum += (data[i]-average)*(data[i]-average);
  }
  return Math.sqrt(sum/data.length);
}
```

(b) Give the worst-case time complexity for the contains() method from your Lookup<Double> class. Explain your reasoning.

(a) For the PureStack<E> interface, list the methods (you do not need to get their names exactly right) and explain what each method does.

(b) For the PureQueue<E> interface, list the methods (you do not need to get their names exactly right) and explain what each method does.

Every integer is evenly divisible by 2^i where $0 \leq i < \infty$, i.e., every integer can be evenly divided by 2 zero or more times. We can determine the value of i by seeing how many times we can divide the integer by 2 and still get an integer result.
Write a recursive method, factor, that returns the value of i. Your method should be a class method.

List the fields required for the inner Node class for a tree data structure.

Precisely and concisely explain why the contains method for a balanced binary search tree is $O(\log n)$.

The class below implements a different version of the hashCode() method. The values returned by the method are integers uniformly distributed between zero and the largest possible integer value in Java, i.e., the probability that the method returns 5 is the same as the probability that the method returns 0, 1, 2, 3, 4, 6, 7, 8, 9, ... or Integer.MAX_VALUE.

```
public class SillyString {

  protected String string;

  // ...

  public int hashCode()
  {
    return Math.random()*Integer.MAX_VALUE;
  }
}
```

Is this a good implementation for hashCode? Why or why not?

(a) Describe how the MorseDecode class makes use of a tree data structure to decode a message in morse code.

(b) Explain what happens in your lab 5 solution if you attempt to encode a character that is not in the morsecode.txt file.