NAME: _____

CS 245 Midterm

Spring 2014

Doodle here.

1. *Solamente Uno*

   Write a class `Set` that manages a collection of unique `String`s. Give it:

   (a) a constructor that constructs the set to be empty,

   (b) a `contains` method that returns a `boolean` indicating whether or not the `Set` contains a `String` given as a parameter,

   (c) an `add` method that adds the `String` given as a parameter to the `Set` if it has not yet been added and does nothing otherwise,

   (d) and a `size` method that returns the number of unique `String`s that have been added to the `Set`.

   Only one instance variable is needed to pull this off: an `ArrayList<String>`, which has identical methods to those described above. However, don't make `Set` a subclass of `ArrayList`. Extending would cause `Set` to inherit methods that would make the `Set` not act like a `Set`.

2. *The n Times*

What is the computational complexity of the following algorithms, in terms of n? Possible answers include $O(1)$, $O(\log_2 n)$, $O(n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$, and $O(n!)$. Briefly justify your answer.

(a) reversing an n-element array

(b) generating all possible pairs of elements in an n-element array

(c) extracting the first three characters from an n-character `String`

(d) coloring all border pixels of an $n \times n$ image black

(e) determining how many times integer n can be divided in half

3. *ExpressionPlus*

   (a) Write an `interface` named `Expression`. It imposes the method `evaluate` on its implementers, which returns an `int`.

   (b) Write a `class` named `AddExpression` that implements the `Expression` interface. Its constructor accepts two `Expression`s as parameters—its left and right operands. Its evaluate method returns the sum of its two operands.

   (c) Write a `class` named `IntExpression` that implements the `Expression` interface. Its constructor accepts an `int` as its only parameter. Its evaluate method returns the `int`.

   (d) Using your classes, construct an expression for $(5 + 6) + 7$.

4. *Buggy Data*

   Considering the following implementation of binary search:

```
int binarySearch(int[] haystack, int start, int end, int needle) {
   if (end < start) {
      return -1;
   }

   int middle = (start + end) / 2;

   if (needle == haystack[middle]) {
      return middle;
   } else if (needle < haystack[middle]) {
      return binarySearch(haystack, start, middle - 1, needle);
   } else {
      return binarySearch(haystack, middle + 1, end, needle);
   }
}
```

   Suppose you feed the following array to binary search. Note that the array is not sorted as the binary search algorithm expects it to be. Which items—if searched for—will still be found?

10	34	40	43	12	37	41	28	46	29	23

5. *All of Them*

   Consider the following class modeling a `Webpage`:

```
public class Webpage {
  private boolean isVisited;
  private String url;

  public Webpage(String url) {
    isVisited = false;
    this.url = url;
  }

  /* Gets a list of the webpages linked from this page. */
  public ArrayList<Webpage> getLinks() {
    // assume this method just works
  }
```

   Add another method to this class that prints out the page's URL and the URLs of all pages reachable from this one, directly or indirectly. Name it `visit`. Each URL must be printed only once, a feat you can achieve by tracking whether or not a page has already been visited. If the page has not yet been visited, print out its URL and visit all the pages to which this page has links.