COLBY COLLEGE
Department of Mathematics
and Computer Science

CS115 **Allen B. Downey**
Spring 1999

Final Exam

Part One: vocabulary (10 points)

1. For a given class of problems, there may be more than one ____________________, although some are more efficient than others.

2. Analyzing the syntactic structure of a formal or natural language is called ____________________.

3. When you ____________________ a variable, you have to specify its type, but when you use it as a ____________________, you don't have to.

4. Name two ways that the elements of an array are different from the instance variables of an object.

5. The Rule of Composition says that any place in a Java program where you can use a variable, you could use any other kind of expression. Give an example of a case where this rule is violated.

Part Two: state diagrams (10 points)

For the following program, draw a state diagram that shows the variables `a`, `jub`, `zoo`, `bob` and the objects or arrays they refer to. What is the output of the program?

```
public class Exam {

  public static int[] make (int n) {
    int[] a = new int[n];

    for (int i=0; i<n; i++) {
      a[i] = i+1;
    }
    return a;
  }

  public static void dub (int[] jub) {
    for (int i=0; i<jub.length; i++) {
      jub[i] *= 2;
    }
  }

  public static int mus (int[] zoo) {
    int fus = 0;
    for (int i=0; i<zoo.length; i++) {
      fus = fus + zoo[i];
    }
    return fus;
  }

  public static void main (String[] args) {
    int[] bob = make (5);
    dub (bob);

    System.out.println (mus (bob));
  }
}
```

Using the terminology we learned in this class, describe what the method named `mus` does and how it does it. Be sure to identify the role of all the variables.

Part Three: debugging (9 points)

The following program contains three errors, including one syntax error, one run-time error, and one logical error. Identify the syntax error and paraphrase the error message it would cause. Identify the run time error and paraphrase the exception it would generate. Identify the logical error and describe the effect it would have on the result.

```
// minmax finds the largest and smallest elements of an
// array.  It creates and returns a new Point object
// that has the largest element stored in x and the
// smallest element stored in y

public static Point minmax (int[] a) {
  int n = a.length();
  Point mm = new Point ();
  mm.x = 0;
  mm.y = 0;

  for (int i=0; i<=n; i++) {
    if (a[i] > mm.x) mm.x = a[i];
    if (a[i] < mm.y) mm.y = a[i];
  }
  return mm;
}
```

Part Four: algorithm design (12 points)

Imagine that there are a large number of people at a party. You would like to know whether any two of them have the same birthday. First, you design a new kind of Java object that will contain birthdays. It contains two instance variables: `month`, which is an integer between 1 and 12, and `day`, which is an integer between 1 and 31.

Next, you create an array of `Birthday` objects that contains one object for each member of the party, and you hire a stupid person to type in all the birthdays.

While the stupid person is busy, you have time to think of an algorithm for checking the array and finding all cases where two (or more) people have the same birthday. Describe your algorithm abstractly, without using Java code.

You should identify the methods you would write and their parameters and return values. Within each method, you should identify any patterns and explain clearly how the method works.

Ideally your algorithm should only traverse the array of `Birthdays` once, but if you cannot think of an algorithm that is so efficient, you can provide a less efficient one.

Part Five: programming (10 points)

One of the common forms of encryption, called RSA, depends on the ability to raise large integers to an integer power. At some point this semester, we saw a (reasonably) fast algorithm for integer exponentiation:

```
// pow raises x to the nth power

public static int pow (int x, int n) {
  if (n==0) return 1;

  // find x to the n/2 recursively
  int t = pow (x, n/2);

  // if n is even, the result is t squared
  // if n is odd, the result is t squared times x

  if (n%2 == 0) {
    return t*t;
  } else {
    return t*t*x;
  }
}
```

The problem with this method is that it will only work if the result is smaller than 2 billion. Rewrite it so that the result is a `BigInteger`. The parameters can still be integers, though.

You can use the BigInteger methods `add` and `multiply`, but you cannot use the built-in `pow` method. On the other hand, you may use the `convertToBig` method described on Homework 9.

Part Six: an easier program (10 points)

Write a method that takes a String as an argument and that returns a boolean indicating whether or not the word contains any double letters.

Your method should be case-insensitive, so that both `Allen` and `Aardvark` yield `true`. You may assume that the String contains no punctuation or spaces.