COLBY COLLEGE
Department of Mathematics
and Computer Science

**CS115** **Allen B. Downey**
**Spring 1999**

**Assignment 5: Recursion**

**Due:** Wednesday 10 March

### 99 Bottles

The first verse of the song "99 Bottles of Beer" is:

> 99 bottles of beer on the wall, 99 bottles of beer, ya' take one down, ya' pass it around, 98 bottles of beer on the wall.

Subsequent verses are identical except that the number of bottles gets smaller by one in each verse, until the last verse:

> No bottles of beer on the wall, no bottles of beer, ya' can't take one down, ya' can't pass it around, 'cause there are no more bottles of beer on the wall!

And then the song (finally) ends.

1. Create a new project called `Bottle` based on the No Graphics stationery. Follow the project-creation instructions carefully!
2. Write a program that prints the entire lyrics of "99 Bottles of Beer." Your program should include a recursive method that does the hard part, but you also might want to write additional methods to separate the major functions of the program.
3. As you are developing your code, you will probably want to test it with a small number of verses, like "3 Bottles of Beer."

### Recurse

1. Create a new project called Recurse, based on the No Graphics stationery.
2. Type in the method from Quiz 6 (it was called `rarify` there, but you should call it `printBinary` now that we know what it does).
3. Write a few lines in `main` that test `printBinary` and run the program. Check that the output is correct.

4. Make a copy of `printBinary` and rename it `printBase`. Generalize it by adding a second parameter called `base` and make it print numbers in any base instead of just base 2.

   HINT: You do not have to understand the algorithm for converting from one base to another in order to do this assignment. One of the things I am suggesting is that generalizing a method can be a mechanical process you can perform without completely understanding the method.

5. The Ackerman function is defined for non-negative integers as follows:

$$A(m,n) = \begin{cases} n+1 & \textbf{if} \quad m=0 \\ A(m-1,1) & \textbf{if} \quad m>0, n=0 \\ A(m-1, A(m,n-1)) & \textbf{if} \quad m>0, n>0 \end{cases} \tag{1}$$

   Write a method called `ack` that takes two `int`s as parameters and that computes and *returns* the Ackerman function. This will be the first non-void method you write; it will not print anything—it will only return an `int`.

6. Test your implementation of Ackerman by invoking it from `main` and printing the return value. Warning: the return value gets very big very quickly. You should try it only for small values of $m$ and $n$.

7. Now that you have the hang of non-void methods, you can write a new version of `printBase` called `convertBase` that *returns* a String containing the result rather than printing the result.

   HINT: You can compose String concatenation with the `return` statement in powerful ways. For example, if you have two pieces of the answer, you can assemble them and return them at the same time:

```
public static String example () {
  String prefix = getPrefix ();
  String suffix = getSuffix ();
  return prefix + suffix;
}
```

**What to turn in**

1. As usual, print your code using a reasonably large font and put your name in the upper-right hand corner of all pages. Please staple the pages together.