COLBY COLLEGE
Computer Science Department

**CS151**
**Fall 1999**

**Allen B. Downey**

## Assignment 5: Recursion

Due: Wednesday 13 October

### Fruitful method

1. Dig out the old `multadd` project and make sure you can still compile and run it.
2. Transform `multadd` into a fruitful method, so that instead of printing a result, it returns it.
3. Everywhere in the program that `multadd` gets invoked, change the invocation so that it stores the result in a variable and/or prints the result.
4. Transform `yikes` in the same way.

### 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.prj` using the `cs151HelloStationery`.
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.

   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.prj`, again based on the `cs151HelloStationery`.

2. 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 value of the Ackerman function.

3. 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 $n$ (not bigger than 3).

**Recurse again**

1. Read Sections 7.1 and 7.2 of the book.

2. Add the following methods to the `Recurse` project.

```
// first: returns the first character of the given String
public static char first (String s) {
    return s.charAt (0);
}

// last: returns a new String that contains all but the
// first letter of the given String
public static String rest (String s) {
    return s.substring (1, s.length());
}

// length: returns the length of the given String
public static int length (String s) {
    return s.length();
}
```

3. Write some code in `main` that tests each of these methods. Make sure they work, and make sure you understand what they do.

4. Write a method called `printString` that takes a String as a parameter and that prints the letters of the String, one on each line. It should be a `void` method.

5. Write a method called `printBackward` that does the same thing as `printString` but that prints the String backwards (one character per line).

6. Write a method called `reverseString` that takes a String as a parameter and that returns a new String as a return value. The new String should contain the same letters as the parameter, but in reverse order. For example, the output of the following code

```
String backwards = reverseString ("Allen Downey");
System.out.println (backwards);
```

should be

```
yenwoD nellA
```

**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.