COLBY COLLEGE
Department of Mathematics
and Computer Science

CS115
Spring 1998

Allen B. Downey

Assignment 8: Strings and Series

Exam question

The following question may seem familiar: one way to evaluate e^x is to use the infinite series expansion

$$e^x = 1 + x + x^2/2! + x^3/3! + x^4/4! + ... \tag{1}$$

If the loop variable is named `i`, then the ith term is equal to $x^i/i!$.

1. Write a method called `myexp` that adds up the first `n` terms of the series shown above. You may use the `factorial` method from the book.
2. You can make this method much more efficient if you realize that in each iteration the numerator of the term is the same as its predecessor multiplied by `x` and the denominator is the same as its predecessor multiplied by `i`. Use this observation to eliminate the calls to `Math.pow` and `factorial`, and check that you still get the same result.
3. Write a method called `check` that takes a single parameter, `x`, and that prints the values of `x`, `Math.exp(x)` and `myexp(x)` for various values of `x`. The output should look something like:

   ```
   1.0      2.708333333333333       2.718281828459045
   ```

   HINT: you can use the String `""` to print a tab character between columns of a table.
4. Vary the number of terms in the series (the second argument that `check` sends to `myexp`) and see the effect on the accuracy of the result. Adjust this value until the estimated value agrees with the "correct" answer when `x` is 1.
5. Write a loop in `main` that invokes `check` with the values 0.1, 1.0, 10.0, and 100.0. How does the accuracy of the result vary as `x` varies? Compare the number of digits of agreement rather than the difference between the actual and estimated values.
6. Add a loop in `main` that checks `myexp` with the values -0.1, -1.0, -10.0, and -100.0. Comment on the accuracy.

Dupledromity

1. Write a recursive method called `isDupledrome` that take a String as an argument and that returns a boolean indicating whether the word is a dupledrome.

   Think about whether there are any helper methods like `first`, `last` and `middle` that will make this program easier to write. What are the preconditions for the helper methods?

2. Test your method extensively to make sure it works for all words, even weird ones like the empty string. Using the Guardian Pattern, see if you can "prove" that it is impossible for your method to cause an exception.

3. Write an iterative version called `isItDupledrome`. Be careful not to get the recursive and iterative versions confused. They will look very different. I find it useful to draw a picture of the string and the indices.

A few more string things

1. Write a method called `printReverse` that takes a String and prints it backwards, all on one line, with a newline at the end.

   This method can be done either iteratively or recursively. I find the iterative version easier.

2. Write a method called `reverse` that takes a String and that returns a String that contains the same letters, but backwards. This method should not print anything. For example, if the argument is `"Allen"`, the return value should be `"nellA"`.

   This method can be done either iteratively or recursively. I find the recursive version easier.

3. The Captain Crunch decoder ring works by taking each letter in a string and adding 13 to it. For example, 'a' becomes 'n' and 'b' becomes 'o'. The letters "wrap around" at the end, so 'z' becomes 'm'.

   Write a method that takes a String and that returns a new String containing the encoded version. You should assume that the String contains upper and lower case letters, and spaces, but no other punctuation. Lower case letters should be tranformed into other lower case letters; upper into upper. You should not encode the spaces.

4. Generalize the Captain Crunch method so that instead of adding 13 to the letters, it adds any given amount. Now you should be able to encode things by adding 13 and decode them by adding -13. Try it.