CMSC 10200: Homework 3

assigned: June 24, 2005; due: Tues, June 28, 2005 (before 5 PM)

The following exercises will give you practice with arrays, iteration and Strings.

Please write the given functions as `static` functions in a program called `HW3.java`. Make sure you write enough tests to convince yourself that all your functions work, and include all such tests in the `main` function.

I have made a framework for `HW3.java`, which you may get from the course website. Feel free to download it and use it as a starting point. Please include your name in a comment at the top of the program.

Save `HW3.java` in a folder called `hw3` and submit it with the command `hwsubmit cs102 hw3`.

- `static boolean allTrue(boolean[] bools)`

  This function should return `true` if every boolean in the given array is true. If the array is empty, return `true`.

- `static boolean anyTrue(boolean[] bools)`

  This function should return `true` if any boolean in the given array is true. If the array is empty, return `false`.

- `static boolean allFalse(boolean[] bools)`

  This function should return `true` if no boolean in the given array is true. Hint: this function need not include a `for` loop directly—it can employ one of the preceding functions.

- `static boolean allLessThan(int bound, int[] ns)`

  This function should return `true` if all integers in `ns` are less than the given value `bound`. If the array is empty, return `true`.

- `static boolean allNegative(int[] ns)`

  This function should return true if all integers in `ns` are negative. Hint: use the previous function.

- `static int selectMin(int[] ns)`

  This function should return the smallest integer in `ns`. If the array `ns` is empty, throw an `IllegalArgumentException`. The minimum need not be unique in the array `ns`.

- `int productFrom(int low, int high)`

  If `low` is larger than `high`, return 1. If `low` is the same as `high`, return `low`. If `low` is less than `high`, return the product of all integers between `low` and `high` inclusive.

- `int occurrencesOf(char c, String s)`

  This function should return the number of occurrences of the character `c` in the String `s`. Some examples:

  ```
  occurrencesOf('l', "Illinois")   -> 2
  occurrencesOf('b', "basketball") -> 2
  occurrencesOf('s', "basketball") -> 1
  ```

  If the String `s` is empty, return 0.

- `double percentageOf(char c, String s)`

  This function should return what percent of the characters in `s` are `c`. Some examples:

  ```
  percentageOf('l', "Illinois")   -> 0.25
  percentageOf('b', "basketball") -> 0.2
  percentageOf('s', "basketball") -> 0.1
  ```

  If the String `s` is empty, return 0.0.

Extra Credit Problem

Consider the following recursive algorithm for reversing a String.

- if the length of a String s is less than two, its reversal is s itself
- if the length of a String s is greater than or equal to two, its reversal is the last character of s concatenated with the reversal of all but the last character in s

Implement the function

```
String reverse(String s)
```

according to this definition.