Homework 10: String Search

Due: See T-Square

Important

There are general homework guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 7**. This includes unused code, so don't submit extra files that don't compile. (Java is backwards compatabile so if it compiles under JDK 6 it *should* compile under JDK 7.)
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. Always be very conscious of efficiency. Even if your method is to be O(n), traversing the structure multiple times is considered non-efficient unless that is absolutely required (and that case is extremely rare). Traversing beyond your data and into unoccupied areas of the backing store is not-efficient and is not actually O(n). See more discussion of Big O and efficiency below.
7. You must submit your source code, the `.java` files, not the compiled `.class` files.
8. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.

String Search

For this assignment you will be coding two different string search algorithms: Boyer-Moore and Rabin-Karp. Using these algorithms you will search for a needle (particular string) in a haystack (body of text) and return all the indicies where a match occurs. A PDF for Boyer-Moore will be provided that contains examples and detailed instructions on the algorithm. A brief description of Rabin-Karp will be provided in the javadocs.

Driver

You will also be required to create a Driver class that will take in a needle from the user and find all matches in a user provided haystack for a given algorithm. You should also provide the runtime for the algorithm that is chosen so you can compare how efficient a string search algorithm is with different alphabets.

Style and Formatting

It is important that your code is functional and is also written clearly and with good style. Efficiency also always matters. Be sure you are using the updated style checker located in T-Square-Resources. (It was updated on February 5, 2014.) Be sure to run your code against this latest style checker. While it is backward compatible with the original style checker, some style rules have been made easier to pass, so it is in your interest to update to this Feb 5 style checker file.) If you feel like what you wrote is in accordance with good style but still sets off the style checker please email Kush Mansingh kmansingh3@gatech.edu with the subject header of "Style XML".

Javadocs

All public methods must have javadocs (or @Override's where appropriate). All private methods must be commented, but it is not necessarily that they be in javadoc style.

Provided

The following files have been provided to you:

1. `StringSearchInterface.java`
2. `BasicStringTests.java`
3. `CheckStyle.zip` - see T-Square Resources for the latest file from February 5, 2014

Deliverables

You must submit all of the following files. Please make sure the filename matches the filenames below.

1. `StringSearch.java`
2. `Driver.java`

Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interface, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc. You may attach each file individually, or submit them in a zip archive.