Important

There are general submission 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 8**. This includes unused code, so don't submit extra files that don't compile.
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. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. Do not add any new instance variables.
8. All methods must be efficient, even if a runtime is not specified.
9. 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 Searching

For this assignment you will be coding 3 different string searching algorithms: Boyer-Moore, Knuth-Morris-Pratt, and Rabin-Karp. There is information about all three in the interface and more information about Boyer-Moore and KMP in the book (also under resources on T-Square).

Do not use `Math.pow` in any method for this assignment.

CharSequence

`CharSequence` is an interface that is implemented by `String`, `StringBuffer`, `StringBuilder` and many others. We have also included a class, `SearchableString`, that implements `CharSequence`. You may use any class that implements `CharSequence` while testing your code. `SearchableString` allows you to see how many times you have called `charAt`. We will be looking at the number of time you call `charAt` while grading.

Do not use any method except `charAt` and `length`; all other methods will either throw an exception or will return invalid data.

Exceptions

When throwing exceptions, you must include a message by passing in a String as a parameter. **The message must be useful and tell the user what went wrong**. "Error", "BAD THING HAPPENED", and "fail" are not good messages.

For example:

```
throw new PDFReadException("Did not read PDF, will lose points.");
throw new IllegalArgumentException("Cannot insert null data into data structure.")
```

Style and Formatting

It is important that your code is not only functional but is also written clearly and with good style. We will be checking your code against a style checker that we are providing. It is located in resources along with instructions on how to use it. We will take off a point for every style error that occurs. If you feel like what you wrote is in accordance with good style but still sets off the style checker please email Jonathan Jemson (jonathanjemson@gatech.edu) with the subject header of "CheckStyle XML".

Javadocs

Javadoc any helper methods you create in a style similar to the Javadocs for the methods in the interface.

Forbidden Statements

You may not use these in your code at any time in CS 1332.

- `break` may only be used in switch-case statements
- `continue`
- `package`
- `System.arrayCopy()`
- `clone()`
- `assert()`
- `Arrays` class
- `Array` class
- `Collections` class
- Reflection APIs

Debug print statements are fine, but should not print anything when we run them. We expect clean runs - printing to the console when we're grading will result in a penalty. If you use these, we will take off points.

Provided

The following files have been provided to you:

1. `StringSearchingInterface.java` This is the interface you will implement. All instructions for what the methods should do and the requirements for each method are in the javadocs. **Do not alter this file.**
2. `StringSearching.java` This is the class in which you will actually implement the interface. Feel free to add private helpers but **do not add any new public methods.**
3. `SearchableString.java` This is the class that we will use to test your searches. It implements `CharSequence`. **Do not alter this file.**
4. `StringSearchingTestsStudent.java` This is the test class that contains a set of tests covering the basic cases for searching. It is not intended to be exhaustive nor guarantee any type of grade. **Write your own tests to ensure you cover all edge cases.**

Deliverables

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

1. `StringSearching.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.