|  |  |
| --- | --- |
| **Project #2** | **CMSC 132** |
| **Due:  Saturday 09/17 at 11:00 pm** | **Object-Oriented Programming II** |
| **Type of project: Open** | **Fall 2016** |

**S N I P P E T**

**Introduction**

When you search for a document on the web, you typically provide the search engine with a list of "terms" to look for, and the search engine responds to your query with a list of results. Each result provides a link to a document containing all of the terms in your query. Many search engines also provide a "snippet" for each result showing the search terms in the context of the document.

For example, a search for the terms:     proposition seven liberty

might return the following result, which includes a "snippet" from the document showing the search terms in context:

Notice that the search terms shown in the snippet are not necessarily in the same order as they were listed in the query.

**What You Will Do**

You will write code that, given a document (a sequence of words) and set of search terms, will find the *minimal length* subsequence in the document that contains all of the search terms. If there are multiple subsequences that have the same minimal length, you may return any one of them.

For example, suppose we are searching for the terms "A", "B", and "C" in the document below:

K W D A I B D W C D W S D B F A C E S D B C D E S A D B X

There are many subsequences of this document that contain all of the search terms. For example:

A I B D W C

C D W S D B F A

B C D E S A

etc.

But, the shortest subsequence that contains all of the search terms is this one:

B F A C

There are many algorithms that will do the job. If we assume that the number of search terms is a constant, can you find an algorithm that runs in time O(n), where n is the number of words in the document? (This is not an official requirement, but if your code is way too slow then it could timeout during our testing, which will be considered a failure of the test(s) in question.)

**MinimumSnippet Class**

This is the class you will write. We have provided you with a skeleton of the class for you to fill in. As you will see, the constructor for the class does most of the work, including locating the minimum snippet containing a list of terms in a given document.

For a complete description of what you must implement in this class, please see the [Javadoc for the MinimumSnippet class.](http://docs.google.com/doc/) Read this document very thoroughly and carefully!

**Getting Started**

To work on a class project, the first thing you must do is to "check out" the project from your CVS repository. This was demonstrated in class, and is also described in the [Eclipse Tutorial.](http://www.cs.umd.edu/eclipse/) The basic steps are:

1. Start Eclipse and go into the CVS perspective. Create a CVS connection to your account.
2. Open the connection by clicking on the little triangle next to it, then click the triangle next to the entry that says "Head" - this should reveal the files that are currently on your CVS repository.
3. Right-click the project called "132Fall2016Project2" and choose "check out".
4. Go back to the Java perspective.

Once you are back in the Java perspective, you should see the project folder on the left, and you should be able to begin working on the project.

**If you write the project from scratch, without checking out the "132Fall2016Project2" files from your CVS repository, you will not be able to submit your work easily.**

**Public Tests**

We have provided several JUnit tests in the class PublicTests.java. You may run these 6 tests as often as you like on your own machine to see if your code is working properly. The results of these tests will also be used by the submit server as a portion of your grade on the project.

**Submitting Your Work**

Submit your project from Eclipse (within Java perspective) by right-clicking the project folder and selecting "submit" . When it asks for your ID and password, be sure to use your University of Maryland "Directory ID" and the corresponding password. You may submit as many times as you want -- we will always grade the submission that is received most recently.   After you have submitted your project, you should visit the [submit server](http://submit.cs.umd.edu/).  There you can obtain limited feedback about how well your project is performing.  The number of times you can run our tests on your project (before the due date) is limited.  **The earlier you begin working on the project, the more opportunities you will have to see how your project performs on our tests before the due date!**

**Grading**

There are 6 public tests and 11 release tests for this project. Together, these tests will comprise 90% of your grade. The remaining 10% will be "style grading", done by visual inspection. See  [this link](http://www.cs.umd.edu/class/fall2016/cmsc132-04XX/content/resources/StyleGuidelines.html)  for a reminder of what elements of style the graders will be considering.

[Web Accessibility](https://www.umd.edu/web-accessibility)