The State Universtiy of New York, Korea
Computer Science

CSE 114 Handout 8: PS 5 April 11, 2019

This problem set is due at **11:55pm on Thursday, April 18, 2019, KST**. Don't go by the due date that you see on Blackboard because it is in EST. Go by the one given in this handout.

- Please read carefully and follow the directions exactly for each problem. Files and classes should be named exactly as directed in the problem (including capitalization!) as this will help with grading.
- You should create your programs using emacs.
- Your programs should be formatted in a way thats readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.
- This problem set assumes you have installed Java and emacs on your computer. Please see the course web for installation instructions.
- You must use the command-line interface to run your programs. That is, you must use the `javac` and `java` commands to do this problem set. Do not use Eclipse yet.
- **Be sure to include** a comment at the top of each file submitted that gives **your name** and **email address**.

Your submission should include the following file. Do not hand in `.class` files or any other that I did not ask for:

```
Pig.java
DictDoc.java
SC.java
```

Problem 1 (50 points)

In the [given] link next to this handout, you will find `Pig.java`, which is an incomplete program. You are asked to complete the program as specified in the given file.

While working on this problem, please pay attention to how the program is designed. This is a good example of a top-down design approach in program design.

Hand in your completely functional `Pig.java`.

Problem 2 (50 points)

In this problem you will build a spell checker program. In the [given] link next to this handout, you will find `DictDoc.java`, which contains a dictionary (named `dict`) in the form of a string array and document to be spell checked (named `doc`). They are each introduced as a static field in `DictDoc.java`.

Your task is to write another class named `SC` (short for `SpellCheck`) in a file named `SC.java`. This class will contain a `main` method that drives the spell checking process. It will spell check `doc` against `dict`. As you will see `dict` only contains a small number of words. So many common English words will not be recognized as spelled correctly, which is okay for our purpose. If you were to do it right, you would include all known English words in `dict` but we are not going to worry about it in this problem. We will do so when we try an improved version of this program later in the semester.

A few things to note. Some words in `doc` are capitalized, e.g., `For`, `Try`, `First`, etc. You will need to convert them into lower-case letters before you check them against the dictionary (`dict`) since the dictionary only contains words in lower-case letters. Another note: Any non-alphabet letters in English should be recognized as separators (delimiters). As for the delimiters, you only need to worry about the ones used in `doc`. Of course you are welcome to include more if you want.

I am giving a sample output of my solution as `SCOutput.txt`. That is, I expect your program to work as mine in terms of the output that it generates.

Hand in your completely functional `SC.java` along with `DictDoc.java` although you won't change the second file.