Lab 5: Dynamic Programming: Part 2.
The "Let's Get it Right" Lab

Due date: Thursday, May 13, in-class.

Lab Assignment

Assignment Preparation

This is a **pair programming** lab. The goal of this lab to give you an opportunity to finish the design and development of the algorithm that you were asked to come up with for the midterm exam.

Each pair will develop, validate, test and submit one copy of the assignment. Each partner in a pair will get the same grade for this lab.

"Is It a Text" Problem

Recall the specification from the midterm exam. You are given a string of n characters $s[1..n]$ which you believe to be a corrupted text document in which all punctuation has vanished (so that it looks something like "thisisatest"). You wish to reconstruct the document using a dictionary, which is available in the form of a Boolean function dict(i,j), which returns for a string s[i..j] the value **true** if it is a dictionary word, and the value **false** otherwise.

Commentary. This is NOT an optimization problem. Rather, this is a **yes/no** problem. The answer we are looking for is either **yes**, if a given string consists of a sequence of valid words, or **no**, if it does not.

However, this problem can also be solved using dynamic programming.

Greedy Algorithms don't Work. The following simple example demonstrates that a greedy approach that can be characterized as follows:

Find the first substring $s[1..k]$ of $s[1..n]$, such that $dict(1, k)$ is **true**.
Then solve the problem recursively for $s[k+1..n]$

DOES NOT WORK!

Consider the following simple dictionary:

```
i
id
idol
red
```

Consider the string `"idolred"`. The greedy algorithm based on the rule above will break the string as follows: `i-dolred`. Since `"dolred"` is NOT in the dictionary, the greedy algorithm will output **false**.

At the same time, this string is a valid text sequence: `idol-red`.

Task

Each team shall develop and implement a dynamic programming algorithm for determining if a given string is a valid sequence of dictionary words.

To make grading straightforward, your solution shall satisfy the following conditions.

1. Create a Java class `TextChecker`. Instances of this class shall have as their instance variables the text string which will be checked, and the list of dictionary words. The choice of the specific data structures for you to use is left up to you.

2. Implement `TextChecker.dict(int i, int j)` method, which will determine if the substring of an input string starting at character i and ending at character j is a dictionary word. This method shall encapsulate the search over the list of words stored in its dictionary instance variable.

3. Implement `TextChecker()` constructor which creates an empty instance.

4. Implement `TextChecker.setDictionary(dict)` method, which takes as input the data structure containing the entire dictionary and initializes it in the instance of the `TextChecker` class.

5. Implement `TextChecker.setString(String s)` method, which sets the input string.

6. Implement `TextChecker.isText()` method, which runs your dynamic programming algorithm to determine if the input string is a sequence of words from the dictionary.

7. Implement `TextChecker.split()` method, which recovers the actual split of the input string into dictionary words *for input strings on which* `TextChecker.isText()` returns **true**. The `split()` method prints the split of the string into dictionary words.

8. Implement a test environment for the `TestChecker` class, in which an instance of the class is created, gets filled with the dictionary and a string, and the string is analyzed. Make sure your code has the ability to read the list of dictionary words from a file (similarly, make sure your code has the ability to read the input string from file).

Sample Domain

Your program shall work on any dictionary and any input text.

For testing purposes, please use the following dictionary

```
i
item
am
sam
ma
red
dare
rare
re
in
into
to
a
the
main
and
an
```

Some test strings you can use (all these are proper word sequences):

```
samiam
iamsam
iamiamiam
maiamsam
iteminred
themainitemandtherarereddare
itemandtheraredare
maintothered
redare
reddare
rareanddare
```

```
redandrareinmain
themaindare
```

And some strings that are not proper word sequences:

```
aandb
intotooandthe
themainsambutnotthelastone
stop
iamtired
redraredareinannitem
```

Deliverables

This part of the lab has both electronic devliverables and an in-person demonstration. All electronic deliverables shall be submitted by the assignment deadline using the following `handin` command:

Use `handin` to submit:

```
$ handin dekhtyar-grader lab05-349 <files>
```

Electronic Deliverables. Submit all the Java files you created for this lab. Submit a README file with the list of people on the team.

Demonstration. You will conduct an in-person demonstration of your solution during the Thursday, May 13 lab. The demonstration does not yield a letter grade, but without it, your submission of Lab 5 is considered incomplete and earns a grade of 0.

Good Luck!