CHARTER PROGRAMMIG CUP 2K17

#LIT

General Rules:

1. DON'T START UNTIL WE TELL YOU
2. When prompted, start with Problem 1 and do not move on until you've had your answered confirmed to be correct by Miles/one of the runners
3. For outside help, you may only consult the official Oracle Java documentation https://docs.oracle.com/javase/7/docs/api/. For outside classes, You may only make imports using **java.util.***, or using any of the classes provided to you in the given directory (namely, In.java, StdIn.java, StdOut.java, and ST.java). You may of course ask for clarification from one of the runners if one of the problems/rules appears unclear.
4. You must use Java. Compile programs with **javac programName.java** and run with **java programName**
5. **Make sure you drink according to the rules below!**

Drinking Rules:

Below, a drink is defined as a mixed drink prepared by our lovely LAMs. When a drink is incurred, both members must drink.

1. At the beginning of the game, each participant will take a drink in proportion to the number of COS courses taken
    a. 2 or fewer courses – No drink
    b. 3-4 courses – 1 drink
    c. 5-6 courses – 2 drinks
    d. 7-8 courses – 3 drinks
    e. More than 8 – 4 drinks

2. The following rules are used to determine the number of courses a person has taken:
    a. Any course with a COS designation counts as 1 course, with the exception of any theory classes
    b. COS classes that involve no programming will not be counted as courses. These are decided on a case-by-case basis. If you're not sure, ask!
    c. The following courses count as two courses: 226, 333, 402, 432
    d. Current courses don't count

3. A drink is incurred for each time the code does not compile. You don't drink for every compiler error – your team just takes 1 drink for your code not compiling. Compiler warnings don't count.

4. A team incurs two drinks (again, that means two drinks per person) by requesting an answer check and getting the answer wrong
5. A team incurs two drinks per runtime error (ArrayIndexOutOfBounds, Stack overflow error, etc.)

6. Every participant takes a drink every 3 minutes.

7. **If either member of a team boots, both members are disqualified from the event.**
    a. **As a corollary, if you feel sick/uncomfortable at any point, you're welcome to switch to water. We want you to turn up in a safe and fun way.**

Do not start unless we tell you to start, motherfucker

This is Lorena

Lorena is a wonderful dancer but struggles sometimes with words. The following problems are related to Lorena's spelling issues. Hopefully, by the end of tonight, we can show Lorena that words don't have to be so hard.

Part 1: Lorena's Nonsensical Words

Lorena tried her hand at spelling some words. Some of the words are great. Others are complete nonsense. We have to tell Lorena which of her words were spelled correctly, and which ones were really badly spelled.

All of Lorena's attempts are in a file named **"nonsense.txt"**. Each spelling attempt is on its own separate line. Starter code for this problem has been written in a file named "**Part1.java**".

For this problem, you must look through the words in "nonsense.txt" – if a word is spelled correctly (i.e. it matches some word in the dictionary), print out "Correct!" If not, print out "Nonsense!"

Stare at this goddamn page till your answers are verified as correct

Part 2: Lorena's (Slightly) Misspelled Words

After her first series of mistakes, Lorena is getting better at spelling! However, things are still not quite perfect. Her most recent attempts at spelling are all almost correct, but are each off by exactly one letter. That is, there is one letter per spelling attempt that is incorrect, and should be another letter.

Note that there can be many correct answers for the same misspelling! For example, observe the attempt "sare" – Lorena could have meant "bare", "care", "dare", etc. She could have also meant "sore". You cannot assume there is exactly one answer per given misspelling.

Your new task is to figure out what Lorena was attempting to spell. You must go through the file "**misspelled.txt"** and determine which dictionary words are exactly one letter off from Lorena's misspelling. All possible answers for a given misspelling must be printed on the same line. If there's no possible way to correct a word, print a blank line.

As with Part 1, starter code has been written for this problem called **Part2.java**.

Almost done! I bet you think you're hot shit. Wait here till your answers are verified.

Part 3: Lorena Forgot the Spaces!

Lorena's spelling is doing so much better, and she's moving on to typing complete sentences! However, she forgot one of the most important things in a sentence – the spaces separating each word. As an example, look at the following line:

"Iamawesome"

With our dictionary, this can be split in two ways:
"I am awesome" or "I am awe some"

All of Lorena's attempts can be found in the file **"nospaces.txt**", with each attempt on its own separate line. You may assume that every example in **"nospaces.txt"** can be turned into a valid sentence with the proper spacing.

Lorena may have also put in sentences with punctuation!
So you can expect lines such as:

"Whoisthebestofficer?Thesocialchairis."

The correct output would be exactly this:
"Who is the best officer? The social chair is."

For each line in nospaces.txt, your code should output all of the valid ways to split the sentence via spaces.

Part3.java contains similar starter code that Part1.java and Part2.java did. We recommend using it.