CS 11 Machine Problem 1 - Simple Word Unscrambler Game Engine

University of the Philippines Diliman

October 08, 2018

Instructions

1. Please solve the following problems before you implement the actual game.
2. You must get a perfect score on this part to get a score on the other parts of the MP.
3. All problems require reading a dictionary of words. See `dictionary_sample.txt` for an example on how the dictionary file is formatted.
4. Please submit on or before November 4, 11:59pm.

Sample Dictionary

You can use `dictionary_sample.txt` to check if your algorithms are already correct.

```
acts
aqua
art
ate
bat
canonists
canvasers
canvasser
cast
cats
colonialists
deposition
eat
eta
fresher
funders
oscillations
penetrances
positioned
rat
repentances
refresh
refunds
retransforms
sanctions
scat
tab
tae
tar
tea
transformers
zoogeographical
```

1 Seeding Words

Requirement #01: When the program starts, it must get the words to be unscrambled from a dictionary that is external to the program. Given the *file name* of the dictionary, the program must be able to load its contents.

1.1 Input (stdin)

Input consists of a string, indicating the file to be loaded.

1.1.1 Sample Input

```
dictionary_sample.txt
```

1.2 Output (stdout)

Output must be the contents of the dictionary, as is.

1.2.1 Sample Output

```
acts
aqua
art
ate
bat
canonists
canvasers
canvasser
cast
cats
colonialists
deposition
eat
eta
fresher
funders
oscillations
penetrances
positioned
rat
repentances
refresh
refunds
retransforms
sanctions
scat
tab
tae
tar
tea
transformers
zoogeographical
```

2 Picking Words

Requirement #02: For every round of the game, the program must pick a word/a set of words to be unscrambled from the dictionary. This leads you to the problem of retrieving a word from a dictionary given its position in the dictionary.

2.1 Input (stdin)

The first line of input consists of a string, indicating the name of the file to be loaded. The next line is the number of test cases t, followed by t test cases. Each test case is a sequence of integers, indicating the position of a word in the dictionary. The first word in the dictionary has position 0, the second word has position 1, and so on.

2.1.1 Sample Input

```
dictionary_sample.txt
3
0 7 14 21 28
3 1 4 1 5 9
2 7 1 8 2 8
```

2.2 Output

For each test case, output each word corresponding to its position in the dictionary. All words must be printed in one line, and each word must be separated by a space character.

2.2.1 Sample Output

```
acts canvasser fresher refresh tar
ate aqua bat aqua canonists cats
art canvasser aqua cast art cast
```

3 Searching for Anagrams

Requirement #03: The player may have an option to choose a game mode. For example, one mode could be to get all anagrams of a certain word by unscrambling a word. This means that for every word in the dictionary, the program must be able to find all the anagrams of the word.

3.1 Input (stdin)

The first line of input consists of a string, indicating the name of the file to be loaded. The next line is the number of test cases t, followed by t test cases. Each test case is a string of characters.

3.1.1 Sample Input

```
dictionary_sample.txt
3
rat
cats
colonialists
```

3.2 Output

For each test case, output all the words that can be generated from the word. All words must be printed in one line, and each word must be separated by a space character. The words must be arranged in lexicographical order. Also include the original word in the output.

3.2.1 Sample Output

```
art rat tar
acts cast cats scat
colonialists oscillations
```

4 Combining Words

Requirement #04: Another game mode could be to get all the words from a random sequence of characters, and not from a word.

To ensure that at least one word can be generated from the random sequence of characters, the random sequence of characters must be generated from one or more words in the dictionary. Given a set of words, one approach to generate a random sequence of characters is by getting the shortest string containing the characters that are required to unscramble each word (for the sequence to be random, the words must be randomly picked by our program). The program must be able to generate the shortest string containing the characters that are required to unscramble a set of words.

4.1 Input (stdin)

The first line of input consists of a string, indicating the name of the file to be loaded. The next line is the number of test cases t, followed by t test cases. Each test case is a set of words separated by a space character.

4.1.1 Sample Input

```
dictionary_sample.txt
3
art acts refresh
repentances transformers
ate refunds deposition
```

4.2 Output

For each test case, output the shortest string containing the characters that are required to generate all words. The characters in the string must be arranged in lexicographical order (note that in the actual game, the characters must be arranged in random order).

4.2.1 Sample Output

```
aceefhrrst
aceeefmnnoprrrsst
adefiinooprstu
```

5 CHECKING WORDS

Requirement #05: From the string of letters that the program generated, the player will get points by entering a word whose letters are found in the string. Note that the word may not be in the set of words that the program picked (along with their anagrams) when generating the string. This means that a player will only get points if the the word that he/she entered contains letters that are found in the string of letters AND the word is found in the dictionary. The program must be able to check the two conditions.

5.1 Input

The first line of input consists of a string, indicating the name of the file to be loaded. The next line is the number of test cases t, followed by t test cases. Each test case contains two strings, separated by a space character. The first string is the generated string of letters. The second string is a word that a player entered.

5.1.1 Sample Input

```
dictionary_sample.txt
9
trfaeercsh refresh
trfaeercsh first
trfaeercsh fresh
rnpcemtreaserfsno transformers
rnpcemtreaserfsno pants
rnpcemtreaserfsno meant
feptoasurnidoi eat
feptoasurnidoi stunned
feptoasurnidoi deposit
```

5.2 Output

For each test case, output `"True"` if the letters of the word is in the generated string and the word is also in the dictionary. Output `"False"` otherwise.

5.2.1 Sample Output

```
True
False
False
True
False
False
True
False
False
```

6 Computing Scores

Requirement #06: When a player unscrambles a word, the program must give points to the user. One way to compute scores is to give one point for every valid word unscrambled, where the total score is the total number of words. Another way is to compute the scrabble points of a word. In this approach, the total score is the sum of the scrabble points of all words. The scrabble point of a word is computed by adding the points corresponding to each letter of the word.

1 point	e	a	i	o	n	r	t	l	s	u
2 points	d	g								
3 points	b	c	m	p						
4 points	f	h	v	w	y					
5 points	k									
8 points	j	x								
10 points	q	z								

The program must be able to compute the scrabble points of all words. Moreover, given a string of letters, the program must be able to compute the highest score that can be achieved by unscrambling words from the string.

6.1 Input

The first line of input consists of a string, indicating the name of the file to be loaded. The next line is the number of test cases t, followed by t test cases. Each test case is a string of letters.

6.1.1 Sample Input

```
dictionary_sample.txt
3
rnpcemtreaserfsno
trfaeercsh
feptoasurnidoi
```

6.2 Output

For each test case, output the highest score that can be achieved by unscrambling words from the string. Use *scrabble points* in computing the highest score.

6.2.1 Sample Output

```
112
74
72
```