Problem 2: Decryption (findBestShift)

Your friend, who is also taking 6.00.1x, is really excited about the program she wrote for Problem 1 of this problem set. She sends you emails, but they're all encrypted with the Caesar cipher!

If you know which shift key she is using, then decrypting her message is an easy task. If the string `message` is the encrypted message and `k` is the shift key she is using, then calling `applyShift(message, 26-k)` returns her original message. Do you see why?

The problem is, you don't know which shift key she is using. The good news is, you know your friend only speaks and writes English words. So if you can write a program to find the decoding that produces the maximum number of real English words, you can probably find the right decoding (there's always a chance that the shift may not be unique. Accounting for this would use statistical methods that we won't require of you.)

pseudocode

Right now, you should take time to write some pseudocode! Think about an algorithm you could use to solve this problem and write the steps down. Then, you can verify your algorithm with the supplied pseudocode in `ps6_pseudo.txt` before coding.

After you've done writing and checking your pseudocode, implement `findBestShift()`. This function takes a wordList and a sample of encrypted text and attempts to find the shift that encoded the text. A simple indication of whether or not the correct shift has been found is if most of the words obtained after a shift are valid words. Note that this only means that *most* of the words obtained are actual words. It is possible to have a message that can be decoded by two separate shifts into different sets of words. While there are various strategies for deciding between ambiguous decryptions, for this problem we are only looking for a simple solution.

To assist you in solving this problem, we have provided a helper function, `isWord(wordList, word)`. This simply determines if word is a valid word according to the wordList. This function ignores capitalization and punctuation.

Hints

Using string.split

You may find the function `string.split` useful for dividing the text up into words.

```
>>> 'Hello world!'.split('o')
['Hell', ' w', 'rld!']
>>> '6.00.1x is pretty fun'.split(' ')
['6.00.1x', 'is', 'pretty', 'fun']
```

Test Cases

```
>>> s = applyShift('Hello, world!', 8)
>>> s
'Pmttw, ewztl!'
>>> findBestShift(wordList, s)
18
>>> applyShift(s, 18)
```

```
'Hello, world!'
```

Test 1: findBestShift(, 'HELlo, world!')

Output:

```
0
```

Test 2: findBestShift(, 'lWt Fjxo xh... wpGs!')

Output:

```
11
```

Test 3: findBestShift(, "Dro dokmROb'C xkwo sc DklsDrK?")

Output:

```
16
```

Test 4: findBestShift(, 'DE, rKj jxuhu yi q JQ dqcut Qblyd!')

Output:

```
10
```

Test 5: findBestShift(, 'dhzo')

Output:

```
19
```

Test 6: findBestShift(, 'kyl jmayj qmKCrfgle gkkClqc KmkCLRypW')

Output:

```
2
```

Test 7: findBestShift(, 'aJan yujwc bDaAxdwM bcanwpcqnw wxkxmh cxMjh')

Output:

```
17
```

Test 8: findBestShift(, 'ziqam cvtmaa lozmm eWzapqx kczZmvb JQBbmz lwkBwz xwaaMaa nmiab Zwcop')

Output:

```
18
```

Test 9: findBestShift(, 'KadwfL uggc Kgewlaewk ljsFkdSlw gofwJkzah lzawx tQ ewjuzsfl kusllwj svnakw dacw eadc Dguc tskAu vwyjww')

Output:

```
8
```

Test 10 (No valid shift available): findBestShift(, 'smgsn tphihm reejgd')

Output:

```
0
```