Final Study Problems - CSC 401

Notes

- these problems were taken from an exam with a different time limit. they may differ in length and complexity from your final.
- the problems are intended to be representative but not comprehensive, they do not necessarily cover all the material covered on the final.

Problems

`sentencesByLength`

Write a function `sentencesByLength` that reads the text in the file filename and **returns** a dictionary where each key represents a number of words, and the corresponding value is a list of those sentences in the file that have that number of words.

For example the file sentences1.txt has two sentences with 9 words, so the returned dictionary will contain an entry with the key 9 and the two sentences of length 9 contained in a list: `{ 9: ['The quick brown fox jumps over the lazy dog.', "There's a message for you if you look up."], ...}`

Guidelines:

- list the lengths and sentences in the order they are encountered
- sentences end with a `'.'`, you may assume that they only occur at the *end* of sentences, never in the middle.
- in the file sentences may continue onto multiple lines. but in your output they should be listed without newline `\n` characters
- do not record any empty sentences
- Note: requires input files sentences1.txt and sentences2.txt

```
>>> sentencesByLength( 'sentences1.txt' )
{9: ['The quick brown fox jumps over the lazy dog.', "There's a message for you
if you look up."], 7: ['Just go ahead and press that button.']}
>>> sentences = sentencesByLength( 'sentences2.txt' )
>>> type(sentences)
<class 'dict'>
>>> sentences[8]
['The crowd yells and screams for more memes.', 'They were excited to see their
first sloth.']
>>> sentences[12]
['I ate a sock because people on the Internet told me to.', "I would have gotten
the promotion, but my attendance wasn't good enough.", 'The random sentence
generator generated a random sentence about a random sentence.']
>>> [ (k, len(sentences[k])) for k in sentences]
[(11, 3), (6, 2), (8, 2), (17, 1), (9, 2), (10, 1), (18, 2), (14, 2), (15, 1),
(12, 3), (20, 2), (13, 3), (16, 1), (7, 2)]
```

`RandomCode`

Write a `class RandomCode` that allows the creation and guessing of a random code of specified length and drawn from a given string of characters.

- Note, that the code **is allowed to have repeated** characters.
- The only "set/modify" method is the constructor `__init__` that creates and remembers a randomly chosen code.
- All other methods are "get" methods, they **return** information about the code that is stored.

Implement the methods specified below:

`__init__`

- create a code with given length and with each character chosen from given characters.
- The given characters are **not** the code, but the characters the code should be chosen from.
- Note that this code should be stored as an attribute of `self` so it can be referred to in other methods.

`__repr__` - **returns** a string containing the actual code that was assigned in `__init__`, see below for format.

```
>>> import random
>>> random.seed(0)
>>> code = RandomCode(4, "ABCDEF") # 4 characters chosen from "ABCDEF"
>>> # show the code that was randomly assigned
>>> code
RandomCode('DDAC')
>>> RandomCode(4, "ABCDEF")
RandomCode('EDDC')
>>> RandomCode(8, "AB") # 8 characters chosen from "AB"
RandomCode('BBAABAAB')
```

`getSecret` - **returns** the code as a str

`getHint` - **returns** a *hint*, that is a string consisting `'?'`s with the same length as the code

```
>>> random.seed(0)
>>> code = RandomCode(4, "ABCDEF")
>>> code
RandomCode('DDAC')
>>> code.getSecret()
'DDAC'
>>> code.getHint()
'????'
>>> code = RandomCode(8,"ABCD")
>>> code
RandomCode('DDCDCBBC')
>>> code.getHint()
'????????'
```

`correct` - given a string guess, **returns** `True` if the guess is correct `False` otherwise.

```
>>> random.seed(0)
>>> code = RandomCode(4, "ABCDEF")
>>> code.correct('DDAC')
True
>>> code.correct('ADAC')
False
>>> code.correct('ADACD')
False
```

`correctPositions` - given a string guess **return** the number of guessed characters that have both the correct position and the correct value.

- In the first case below the answer is 2 because the guess "**D**AD**C**" and the code "**D**DA**C**" have the same characters in exactly 2 positions.
- In the second case, although the characters in the guess "ACDD" agree with the characters in the stored code "DDAC" , none are in the correct position, so the answer is 0.
- You may **assume** (do not need to check) that the guess has the same length as the stored code.
- If you can't get it quite right, do your best. And make sure that it **returns** an integer in any case. For example, if you have no idea, just make the body `return 0`. That will prevent issues for the next problem, `guessCode`. (You will be counted off here, but not there)

```
>>> random.seed(0)
>>> code = RandomCode(4, "ABCDEF")
>>> code
RandomCode('DDAC')
>>> code.correctPositions("DADC")
2
>>> code.correctPositions("ACDD")
0
>>> [ (guess,code.correctPositions(guess)) for guess in ['EDDC', 'DCEB', 'EBCB',
'AECE', 'FEBC', 'AFAF', 'CDEA', 'CDCE', 'FBED', 'DECA']]
[('EDDC', 2), ('DCEB', 1), ('EBCB', 0), ('AECE', 0), ('FEBC', 1), ('AFAF', 1),
('CDEA', 1), ('CDCE', 1), ('FBED', 0), ('DECA', 1)]
```

`correctLetters` - given a guess string **return** the number of characters that agree, regardless of their position!

- This may be a bit tricky to get right, think about how you would do it by hand.
- If you can't get it quite right, do your best. And make sure that it **returns** an integer. For example, if you have no idea, just make the body `return 0`. That will prevent issues for the next problem `guessCode`. (You will be counted off here, but not there)
- cite sources if used!

```
>>> random.seed(0)
>>> code = RandomCode(4, "ABCDEF")
>>> code
RandomCode('DDAC')
>>> code.correctLetters("ACAD")
3
>>> code.correctLetters("ACDD")
4
>>> [ (guess,code.correctLetters(guess)) for guess in ['EDDC', 'DCEB', 'EBCB',
'AECE', 'FEBC', 'AFAF', 'CDEA', 'CDCE', 'FBED', 'DECA']]
[('EDDC', 3), ('DCEB', 2), ('EBCB', 1), ('AECE', 2), ('FEBC', 1), ('AFAF', 1),
('CDEA', 3), ('CDCE', 2), ('FBED', 1), ('DECA', 3)]
```

`guessCode` (25 pts)

Write a function `guessCode` that implements a simple game that allows a user to guess a `RandomCode`. Details:

- `guessCode` is a standalone function (not inside the class)
- `guessCode` accepts two arguments, the length of the code and a string of characters, used to generate the code it then creates a `RandomCode` object using the given arguments as inputs it then allows the user to repeatedly guess the code until it is correctly guessed or the user types `'QUIT'`.
- After each guess a **print** statement will show the user how many correct letters and how many correct positions there are.
- You should be calling the `correctLetters` and `correctPositions` methods for the `RandomCode` object you created. Note: *this* problem will be graded based on whether they are being called correctly, not that they are implemented correctly.

See the samples below for the user interaction, messages and formatting that are **printed** at each stage.

```
>>> import random
>>> random.seed(7)
>>> guessCode(6,"AB")   # guess random code of length 6 using chars in 'AB'
Try to guess my secret code: ?????? (QUIT to stop)

What is your guess?: AAAAAA
Your guess has 4 correct letters, 4 in the correct position.

What is your guess?: BBAAAA
Your guess has 6 correct letters, 4 in the correct position.

What is your guess?: BABAAA
Congratulations, you got it in 3 guesses!
>>>

>>> random.seed(8)
```

```
>>> guessCode(4,"ABCDEF")
Try to guess my secret code: ???? (QUIT to stop)

What is your guess?: AAAA
Your guess has 0 correct letters, 0 in the correct position.

What is your guess?: BBBB
Your guess has 2 correct letters, 2 in the correct position.

What is your guess?: BBCC
Your guess has 3 correct letters, 1 in the correct position.

What is your guess?: CBBD
Your guess has 4 correct letters, 0 in the correct position.

What is your guess?: BCDB
Congratulations, you got it in 5 guesses!
>>>

>>> guessCode(6,"ABCDEF")
Try to guess my secret code: ?????? (QUIT to stop)

What is your guess?: AABBCC
Your guess has 3 correct letters, 1 in the correct position.

What is your guess?: DDEEFF
Your guess has 2 correct letters, 1 in the correct position.

What is your guess?: ABCDEF
Your guess has 4 correct letters, 1 in the correct position.

What is your guess?: QUIT

You gave up after 3 guesses.
The secret code is EDCCCB.
>>>
```

Ignore everything below. The following is used for the doctest (and make sure it reads user inputs from a file.) **Ignore** this when writing your code, and refer instead to the sample **above**.

```
# DO NOT MODIFY
>>> import sys
>>> si = sys.stdin
>>> sys.stdin = open('inputs.txt')

>>> random.seed(7)
>>> guessCode(6,"AB")
Try to guess my secret code: ?????? (QUIT to stop)
<BLANKLINE>
What is your guess?: Your guess has 4 correct letters, 4 in the correct
position.
<BLANKLINE>
What is your guess?: Your guess has 6 correct letters, 4 in the correct
position.
<BLANKLINE>
What is your guess?: Congratulations, you got it in 3 guesses!
```

```
>>>

>>> random.seed(8)
>>> guessCode(4,"ABCDEF")
Try to guess my secret code: ???? (QUIT to stop)
<BLANKLINE>
What is your guess?: Your guess has 0 correct letters, 0 in the correct
position.
<BLANKLINE>
What is your guess?: Your guess has 2 correct letters, 2 in the correct
position.
<BLANKLINE>
What is your guess?: Your guess has 3 correct letters, 1 in the correct
position.
<BLANKLINE>
What is your guess?: Your guess has 4 correct letters, 0 in the correct
position.
<BLANKLINE>
What is your guess?: Congratulations, you got it in 5 guesses!
>>>

>>> random.seed(17)
>>> guessCode(6,"ABCDEF")
Try to guess my secret code: ?????? (QUIT to stop)
<BLANKLINE>
What is your guess?: Your guess has 3 correct letters, 1 in the correct
position.
<BLANKLINE>
What is your guess?: Your guess has 2 correct letters, 1 in the correct
position.
<BLANKLINE>
What is your guess?: Your guess has 4 correct letters, 1 in the correct
position.
<BLANKLINE>
What is your guess?:
You gave up after 3 guesses.
The secret code is EDCCCB.
>>>

# DO NOT MODIFY
>>> sys.stdin = si
```

Final Study Problems - CSC 401

- Notes
- Problems
  - `sentencesByLength`
  - `RandomCode`
  - `guessCode` (25 pts)