Week 5

Write and run your programs with IDLE editor. Submit finished programs to CodeGrade. Note that some tasks have several steps (A, B, C, …) in CodeGrade.

IMPORTANT: End each input-command string with a newline symbol \n. For example:
`variable = input("Some text:\n")`

Task 1: Write a function `occurrences(ch, string)` which **returns** how many times the character `ch` appears in the string `string`. Your function needs to utilize a loop in which `ch` is compared with each character in `string`, updating the counter accordingly.

Write a main program which asks for a character and a string and then calls the function `occurrences()` with these parameters. The main program should also check that `ch` is just one character. If not, then it reports an error. See examples below.

Example run 1:

```
Enter a character:
ab
Error: Give a single character.
```

Example run 2:

```
Enter a character:
z
Enter a string:
Lahti
The character 'z' appears 0 time(s) in the string.
```

Example run 3:

```
Enter a character:
a
Enter a string:
abrakadabra
The character 'a' appears 5 time(s) in the string.
```

Task 2: Write a function `count_words(sentence)` that takes `sentence` as a paramenter and **returns** the number of words in the sentence. Your function should use a loop which goes through each character in the sentence. It increments the word count whenever it spots a beginning of a new word.

Write a main function which asks for a sentence and calls the the function with this sentence. Main program prints the number of words according to the example below.

Example run 1:

```
Give a sentence:
The quick brown fox jumps over the lazy dog
This sentence contains 9 words.
```

Task 3: An anagram is a word formed by rearranging the letters of a different word, using all the original letters exactly once.

For example:

- `ABBA` and `BAAB` are anagrams.
- `elvis` and `lives` are anagrams.

Write a function `anagram(A,B)` that with two parameters `A` and `B` decides whether `A` and `B` are anagrams. To confirm this, these conditions need to be met:

- There is an equal number of characters in A and B
- For every character found in A, there is an equal number of same characters in B.

The main program should prompt the user to ask for (two strings, and call the function `anagram()` with these words. You may assume that these strings do not contain whitespaces. Depending on the returned results, it should print the answers according to the examples below.

Hint: You can use the function `occurrences()` of Task 1.

Example run 1:

```
Enter string A:
astronomer
Enter string B:
moonstarer
astronomer and moonstarer are anagrams
```

Example run 2:

```
Enter string A:
funeral
Enter string B:
earlrun
funeral and earlrun are not anagrams
```

Task 4: Write a function that takes a string as input and returns a compressed version of the string. For example, for the input "AAAAAAAAAAAAAHHHEEM" (19 characters), the function should return "A13H3E2M" (8 characters). The idea is that if a letter X appears N times in a row, this is compressed as XN. Note that it is “sensible” to add the count only if it's greater than 1, otherwise the text will get longer.

Compute also the compression ratio, that is, the length of compressed text divided by the length of the original text. Round the ratio to two decimals using `round()` function.

Example run 1:

```
Give a string to compress:
python
Compressed string: python
Compressing ratio 1.0
```

Example run 2:

```
Give a string to compress:
aaaabbbbbbbcddddddddeekkkkkk
Compressed string: a4b7cd9e2k7
Compressing ratio 0.37
```

Task 5: Write a function that takes two strings as parameters and returns the Boolean value `True` if the first string is entirely contained within the second, and `False` otherwise. The solution should be based on nested loops, the use of the operator `in` is not allowed in testing.

Write also a main program that prompts the user to enter two strings and then calls this function to determine whether the first string can be found within the second string.

Example run 1:

```
Enter the first string:
part
Enter the second string:
apartment
The first string can be found in the second string.
```

Example run 2:

```
Enter the first string:
tea
Enter the second string:
treat
The first string cannot be found in the second string.
```