1. Perkovic 4.17

4.17 Translate each line into a Python statement using appropriate string methods:
(a) Assign to variable message the string 'The secret of this message is that it is secret.'
(b) Assign to string length the length of string message, using operator len().
(c) Assign to variable count the number of times the substring 'secret' appears in string message, using string method count().
(d) Assign to variable censored a copy of string message with every occurrence of substring 'secret' replaced by 'xxxxxx', using string method replace().

2. Write a function, camelCaseIt(s) that inputs a string that consists of words separated by spaces, and returns a string with no spaces, but the first letter of each word capitalized. Hint: you may want to use the string methods: split, capitalize, and join. But you can do this anyway you prefer. Example input and output:

```python
>>> camelCaseIt('how did we get here')
'HowDidWeGetHere'
>>> camelCaseIt('i am sam sam i am')
'IAmSamSamIAm'
```
3. Observe the accumulator pattern for strings in the following function,

```python
def addSpacesBetweenChars(s):
    s2 = "" # initialize s2 as an empty string
    for ch in s:
        s2 = s2 + ch + ' ' # concatenate previous s2 to ch and space character
    return s2[:-1] # remove final space before returning
```

Test this function and make sure you understand how it works. Strings are similar to lists and ranges, in that they are ordered sequences and may be iterated through with a for-loop. Now, do Perkovic 4.24:

4.24 Implement function `cheer()` that takes as input a team name (as a string) and prints a cheer as shown:

```
>>> cheer('Huskies')
How do you spell winner?
I know, I know!
 Huskies!
And that's how you spell winner!
Go Huskies!
```

4. Write a function `createAlphabet()` that returns ‘abcdefghijklmnopqrstuvwxyz’ by using a string accumulator and a for-loop over the ord values of the lower case letters. No credit for just returning the string by itself.
5. Write a function `myUppercase(s)` that returns the uppercase of string `s` equivalently to `s.upper()`, but instead using a string accumulator, a for-loop over `s` and a check of each character’s ord value. If the ord value indicates that the character is lowercase, then concatenate the corresponding uppercase character by noting that uppercase characters occur exactly 32 below their corresponding lowercase character in the ASCII ord table.