Frame Me! – Recitation Assignment
(Due @ 11:55pm on T-Square, the night of your recitation)

For this Recitation Assignment, you will be developing a name framer. Upon completing this assignment, you should be able to take a given name, let's say, “Roger Federer” and produce the following output:

```
**********
*  Roger Federer  *
**********
```

Pretty neat, huh? While this may seem like a trivial exercise, there is a bit of work that needs to be done.

In particular, we are requiring you to write three functions:

1. frameMe – this is our main function and should take in two parameters: a character and a buffer length.
   Example Function Call: frameMe(“”, 3)*

2. createLine – this is a helper function that will return a string. It will also take in two parameters: a length for the line, and the character being used for the frame.
   Example Function Call: createLine(20, “”)*

3. centerWord – this is a helper function that will return a string. It will take in three parameters: the name to print, the buffer needed for both sides, and the character being used for the frame.
   Example Function Call: centerWord(“Roger Federer”, 3, “”)*

Part One – frameMe function

The first parameter represents the *character* we want to create our frame with (in the above example, the character chosen was “*”). The second parameter will be a *spacing buffer*. If you look closely in the example above, the name Roger Federer is spaced in such a way that it is in the middle of the frame. This notion of a spacing buffer allows you to explicitly state how much space you want between the edge of the frame and the name you are framing.

Within this function you need to query the user for a name that they would like to frame. Do this using the *raw_input()* function.

Next, you need to find some sort of way to determine the length of a single line in our frame. For instance, in the example above, the name 'Roger Federer' is 13 characters long. A buffer of three spaces was also chosen for this particular example (it was passed as a parameter). Hence, the length of one single line is 19 characters long (since there are three spaces to the left of the name, and three spaces to the right). It is up to you to find the proper equation for generating the right length of a line for a particular name and particular buffer.
Hint: the built-in *len()* function will be useful.

Assignment continued on next pg...

Part Two – createLine function
Now that you've been able to determine the length of a single line, go ahead and write a function that *returns* a line of that length with the given frame character
(hint: you will have to pass a parameter given in *frameMe()* to this *createLine()* function.)

> Assuming the *createLine()* function has been implemented correctly, calling *createLine(7, "*")* on its own should return '*******'

Part Three – centerWord function
The *centerWord()* function is a bit tricky-- it requires a bit of conceptual thought. The three parameters are the name that needs to be centered, the buffer that is being used, and the character being used as a frame (hint: again, you will have to pass parameters given in the *frameMe()* function). The function will return a string with the proper buffering...BUT the leading and ending characters of the return line must not be spaces, but rather, must be the specific character provided in the parameter.

To be perfectly clear, look at the following example:

> *centerWord("Sahhhm", 3, "*")* should return *'* Sahhhm *'*
> ***Note: The number of spaces between the * and the letter S is only two. The same is true with the letter m and the ending * – only two spaces. Hence, the leading and ending characters alter the buffer in an aesthetic way. Furthermore, you may assume that when testing your code we will not put a buffer size of less than 2.***

Part Four – putting it all together
Now the easy part! Go back to the *frameMe()* function. You now need to print the proper calls to *createLine()* and *centerWord()* to make your framing complete! (hint: do not make this part difficult. It should be relatively straightforward when looking at the examples provided.)

General Note:
A skeleton file has been provided to help you with this assignment. It is strongly suggested that you type your code in this file, and ensure that all the provided samples inside this skeleton file work.

...End of Recitation Assignment

Written By: Sam Asghari, Spring 2010