CS005 Lab 8: Before you begin, Read the lecture slides again!

Part 1:

Write a function that takes in a string, and returns true if it begins and ends with the same letter.

>> BookEndString('sausage balls') ans = 1	>> BookEndString('ball of fire') ans = 0

Part 2:

Your DNA consist of a string of length 3.1 billion base-pairs. The base-pairs come in 4 types, known as 'A', 'C', 'T', 'G'. So a piece of human DNA (from chromosome 2) looks like this:

GTCAATGGCCAGGATATTAGAACAGTACTCTGTGAACCCTATTTATGGTGGCACCCCTTAGACTAAGATAACAC..

Let us write a function that searches a DNA string for 'A's

```
function Loc = SearchDNA1(DNAString)
 for i = 1 : length(DNAString)
   if DNAString(i) == 'A'
      disp(['found an A at ', num2str(i)])
   end
 end
```

```
>> SearchDNA1('GCAGGATATTAGAA')
found an A at 3
found an A at 6
found an A at 8
found an A at 11
found an A at 13
found an A at 14
```

Let use write a function that searches a DNA string for 'AT's. This is a little tricky, but not too hard.

The trick is to realize that 'A' = 'A' returns true ('1'), and 'A' = 'G' returns false ('0'), and

that 'AT' = 'AT' returns true ('1'), and 'AT' = 'AG' returns false ('0')

So we just need to scan across the string, but stop at the length of the string, minus one. Why "minus one"? Because when we are at the i^{th} location, we are going to look at the next location to the right (as in this: DNAString(i:i+1)) So we are asking this: Do the i^{th} and i^{th} + 1 locations spell 'AT'?

Carefully look at the code below and see what difference it has to the last code...

```
function Loc = SearchDNA2(DNAString)
 for i = 1 : length(DNAString) - 1   % Note the -1 here
   if DNAString(i:i+1) == 'AT'
      disp(['found an AT at ', num2str(i)])
   end
 end
```

```
>> SearchDNA2('GCAGGATATTAGAA')
found an AT at 6
found an AT at 8
```

English sentences end with a full stop '.' However we know that DNA sentences end with what is called a *stop codon*, which happens to be 'TAG'.

Write a function that searches a DNA string for a stop codon, and returns the location (the place where the first letter is.

Hint, I have written 95% of it above.

Part 3:

The most common bigrams in English are “th”, “he”, “in” etc (see http://en.wikipedia.org/wiki/Bigram)

Write a function that takes in a string and returns the bigram frequency of “th”

There are 14 characters (not counting spaces), and “th” appears twice, so 2/14 is 0.1429	>> UseBetterNameThatThis('The man said that') ans = 0.1429

Hint:

To find the number of characters not counting spaces, you could use the DeBlank function we wrote, length(DeBlank(EamonnString)).

Make sure to fix the problem of the mixture of upper and lower case letters.

Part 4:

Write a function that takes in a string, and returns the same string, but it is prepended and appended by three “#”s, and the original letters are now all upper case, and have a “*” in-between them

>> UseBetterNameThatThis('BOB') ans = ###B*O*B###	>> UseBetterNameThatThis('eamonn') ans = ###E*A*M*O*N*N###

Hint:

Don’t try to do this all at once.

Reread the slides, we have done all the parts of this, several times.

In your function, start with an empty string...

```
S2return =[];
```

Run thru a loop of the input string, from i to the length of the string, and each time thru the loop, concatenate the new “stuff” to the S2return, something like this..

```
S2return = [ S2return , InputString(i) , '*')]
```