ECEG 201 – Homework 07

Due at noon on 2020-03-06

This homework assignment asks you to write two Python functions. You must send these two functions to the instructor as two separate source files attached to an email message. The source files should contain **only** the functions requested.

In addition to functionality, your code will be graded on style and format. You should follow the guidelines in PEP 8 (up to *Designing for inheritance*) and remember that you can run `pylint` to check your code. Don't use a `while` when a `for...range` will do. Note the proper naming conventions for variables, constants, and functions...`pylint` assumes that every variable declared outside of a function is a constant, which is not necessarily the case.

1. Write a Python 3 function named *build_message* that takes two byte strings as its parameters. The function should return a byte string. The returned byte string must consist of the first parameter, followed by two binary bytes, followed by the second parameter. The value of the two binary bytes represent a 16-bit integer that is the number of bytes in the second parameter.

   For example:

```
>>> build_message(b'spam', b'sausage')
b'spam\x00\x07sausage'
>>> long =  30 * b" spam"
>>> build_message(b'eggs and ', long)
b'eggs and \x00\x96 spam spam spam spam spam spam spam spam spam spam spam spam
 ↪  spam spam spam spam spam spam spam spam spam spam spam spam spam spam spam
 ↪  spam spam spam'
>>> long = b"sausage spam eggs beans sausage spam spam spam"
>>> really_long = long + long + long + long + long + long + long
>>> len(really_long)
322
>>> build_message(b'hello', really_long)
b'hello\x01Bsausage spam eggs beans sausage spam spam spamsausage spam eggs beans
 ↪  sausage spam spam spamsausage spam eggs beans sausage spam spam spamsausage
 ↪  spam eggs beans sausage spam spam spamsausage spam eggs beans sausage spam
 ↪  spam spamsausage spam eggs beans sausage spam spam spamsausage spam eggs beans
 ↪  sausage spam spam spam'
```

   Note in the last example that the integer 322 will be converted to two bytes: `\x01` and `\x42`, but `\x42` is usually interpreted as the capital letter 'B'.

2. Write a Python 3 function named *count_substrings* that takes two byte strings as its parameters. The function should return an integer that is the number of times that the first parameter appears as a substring in the second parameter, including possible overlaps.

   For example:

```
>>> count_substrings(b'spam', b'spam, spam, eggs, and spam')
3
>>> count_substrings(b'a', b'banana')
3
>>> count_substrings(b'ana', b'banana')
2
```