**CS 299 Project #2 (60 points)**

1. Substitution Cipher. The substitution cipher problem is described in Section 3.5 of textbook and also explained during the lecture. Specific requirements are:
2. Write a function substitutionEncrypt that takes a plaintext and a key to generate a ciphertext.
3. Write a function substitutionDecrypt that takes a ciphertext and a key to generate a plaintext.
4. Write a function keyGen that randomly generates a key.
5. Write a testDrive function that takes a variable-length of strings (must use variable-length parameters), a key (the default value is “bpzhgocvjdqswkimlutneryaxf”, must use default parameter), and a tag (the default tag is ‘E’ which refers to encrypt and the other possible tag is ‘D’ which refers to decrypt.)
6. Test the following cases (use default parameter(s) whenever possible) and print out the original strings, key used, as well as encrypted/decrypted strings.

Test 1: two strings “flow” and “substitutioncipher”, default key, and tag is ‘E’.

Test 2: same as Test 1 except key generated by keyGen

Test 3: three strings “osiy”, “obzy”, “doedlugvusu”, default key, tag is ‘D’

Test 4: same as Test 3 except key generated by keyGen

1. US postal bar code. Encode the zip code and use Turtle to print out the barcode.

The following table shows the encoding for decimal digits. Check sum is (10-sum mod 10).

|  |  |  |
| --- | --- | --- |
| **Digit** | **7 4 2 1 0** | **Barcode** |
| **0** | 1 1 0 0 0 | [POSTNET 0.svg](https://en.wikipedia.org/wiki/File:POSTNET_0.svg) |
| **1** | 0 0 0 1 1 | [POSTNET 1.svg](https://en.wikipedia.org/wiki/File:POSTNET_1.svg) |
| **2** | 0 0 1 0 1 | [POSTNET 2.svg](https://en.wikipedia.org/wiki/File:POSTNET_2.svg) |
| **3** | 0 0 1 1 0 | [POSTNET 3.svg](https://en.wikipedia.org/wiki/File:POSTNET_3.svg) |
| **4** | 0 1 0 0 1 | [POSTNET 4.svg](https://en.wikipedia.org/wiki/File:POSTNET_4.svg) |
| **5** | 0 1 0 1 0 | [POSTNET 5.svg](https://en.wikipedia.org/wiki/File:POSTNET_5.svg) |
| **6** | 0 1 1 0 0 | [POSTNET 6.svg](https://en.wikipedia.org/wiki/File:POSTNET_6.svg) |
| **7** | 1 0 0 0 1 | [POSTNET 7.svg](https://en.wikipedia.org/wiki/File:POSTNET_7.svg) |
| **8** | 1 0 0 1 0 | [POSTNET 8.svg](https://en.wikipedia.org/wiki/File:POSTNET_8.svg) |
| **9** | 1 0 1 0 0 | [POSTNET 9.svg](https://en.wikipedia.org/wiki/File:POSTNET_9.svg) |
| **Start/Stop** | − − − − 1 | [POSTNET BAR.svg](https://en.wikipedia.org/wiki/File:POSTNET_BAR.svg) |

Example:

The ZIP+4 of **55555-1237** yields a check digit of **2** for encoded data of **5555512372**  
Together with the initial and terminal frame bars, this would be represented as:

[POSTNET BAR.svg](https://en.wikipedia.org/wiki/File:POSTNET_BAR.svg)[POSTNET 5.svgPOSTNET 5.svgPOSTNET 5.svgPOSTNET 5.svgPOSTNET 5.svg](https://en.wikipedia.org/wiki/File:POSTNET_5.svg)[POSTNET 1.svg](https://en.wikipedia.org/wiki/File:POSTNET_1.svg)[POSTNET 2.svg](https://en.wikipedia.org/wiki/File:POSTNET_2.svg)[POSTNET 3.svg](https://en.wikipedia.org/wiki/File:POSTNET_3.svg)[POSTNET 7.svg](https://en.wikipedia.org/wiki/File:POSTNET_7.svg)[POSTNET 2.svg](https://en.wikipedia.org/wiki/File:POSTNET_2.svg)[POSTNET BAR.png](https://en.wikipedia.org/wiki/File:POSTNET_BAR.png)

Testing: print out the barcode for: 55555-1237 91768-1234 20500-0000

**Submission instruction:**

For problem 1 save the code in one file (file name: LastnameFirstInitialP2-1.py), cut and paste test run results as comments to the end of source code. For Problem 2 you may need to save codes in two files. Use file name as LastnameFirstInitialP2-2a.py and LastnameFirstInitialP2-2b.py. Upload all files to blackboard along project #2 link.