# Aiden Lab Predoctoral Fellows Application 2017

## Round II (Challenge 1B)

### Encryption Algorithm

At the core of the solution to this problem is an encryption algorithm known as the Markov Chain Metropolis-Hastings algorithm.

Initially a book is chosen as the corpus and a transition matrix of letters is created from the book. Then, the 10-20 digit number is chopped up randomly into smaller pieces each of which has at most the same number of digits as the number representing the number of sentences in the book. For example, the number 1234567899 can be split into 123, 456, 7899 or 1, 234, 5678, 99, both of which are possible results. Suppose for the purposes of this example, the individual pieces are 123, 456, 7899. Then we take the 123$^{rd}$ to 132$^{nd}$ sentences (10 sentences) from the book and apply the Markov Chain Metropolis-Hastings encryption algorithm to it. The decryption algorithm is described at http://probability.ca/jeff/ftpdir/decipherart.pdf. We reverse engineer the algorithm to make it an encryption algorithm. We use this encryption algorithm to encrypt the passage – call this the text A.

Then we encode the encryption key as a string as follows. Suppose we have an encryption key represented as a list of numbers 0-26. The numbers 0-25 represent alphabets and 26 represents space. Now if the number 0 is found at index 25 (assume lists are 0-indexed) then the letter 'z' (index 25) is encrypted into 'a' (number 0). Now we add the length of the string representing the name of the book used for encryption to all the numbers in the encryption key list. This shifted list is encoded by adding a random alphabetic character after every element except the last one. Call this encryption key string, text B.

Now we concatenate texts A and B to get to text C. We repeat the process above with all other pieces (in this case 456 and 7899) and then concatenate all these strings together to arrive at the cipher text.

### Decryption algorithm

For decryption, we retrieve the encryption key string for each passage and build the encryption key from the string. We reverse engineer the encryption key to arrive at the decryption key by switching the roles of index and element in the list. Then we apply each decryption key to its respective passage to get to the original plain text. We then systematically check all sentences in the corpus against this passage to find out the sentence number that refers to this part of the original 10-20 digit number. Finally, we repeat this for all decrypted passages to arrive at the original 10-20 digit number.

**How to run**

Run the main.py file from a Python IDE. It will prompt you for the 10-20 digit number. Once you enter the number, the program runs for a while (it can be as long as 320 seconds) and saves your encrypted cipher text in a .txt file as well as a .json file. These files are saved in the same directory as the main.py file is saved in.

**Note:** You need to install the json python package before you can run the program. Also a few books have been provided – 'bible.txt' and 'war_and_peace.txt'. Both have been downloaded from the open source website https://www.gutenberg.org/ , where more such books can be found.