1: Thisisthebiggestthisclasshaseverbeen -> “This is the biggest this class has ever been”. Since this type of code is encrypted using a Caesar cipher, I had decided to use the brute force method of repeatedly shifting the set of letters down by one until it made a logical sentence. This method did take me 14 tries, but the worst possibility is that it would have taken me 25 tries.

2: rightnowwehaveonlytwotashopefullyifindathirdsoon -> “Right now we have only two **t** as hopefully I find a third soon”. Question two is like question one in that the cipher is a Caesar cipher, so I employed the same method of brute forcing the decryption process by repeatedly shifting the set of letters down by one until it created a logical sentence. I appear to have an extra t from one of the letters in the middle of the cipher which is odd though.

3: ucfispartofthebigtwelvenowdidyouknowthebigtwelvehasfourteenschoolsweirdhuhithoughtso -> “UCF is part of the big twelve now did you know the big twelve has fourteen schools weird huh I thought so”. So, number three was missing the encryption values of a and b meaning that solving the affine cipher by hand would take an extremely long time for me to do by hand, considering there’s a decent sum of outcomes, so I had instead decided to code up a Java program to do it for me. The attached ‘AffineDecryptor.java’ file will be used to explain the code that I had made to solve this question.

4: QAZEGKAJGENMRJGWNIRHMEZIYYHGRAMGULSVPTSWRNSXAYQHSZAPGSVWMGUNHGKAQGNO. The original text provided for question 4 can be encrypted using the Affine formula of (Ax+b) mod 26, which is how I got the above encrypted text. To solve this question, I had decided to modify the Java program so that it can handle input of ‘A’ and ‘B’ while properly outputting the correct encryption. This required some reworking of certain functions which can be seen in the attached ‘AffineEncryptor.java’ file.