1. Introduction  
   The objective of this lab was to create code to Encrypt, Decrypt, and Statistically Attack the Caesar Cipher. This is intended to provide first hand experience with the basic gist of cryptography algorithms. Within the constraints of most programming languages, the shifting of letters in the alphabet cannot directly be accomplished so this assignment provides a challenge to numerically equate letters in order to shift.
2. Procedure/Discussion  
   (Program used is MATLAB)  
   Encrypt: As a general procedure, the inputted string is always converted into a char array for manipulation at the top. An empty string is also declared as the final output, and the length of the input string is assigned to an integer variable for the loop. A main for loop is used to iterate through each letter and individually shift by the key input. The shifted resulting character is added to the end of the output string. The actual process of the shift involves creating a char that is checked to see if it is a letter or space. Spaces simply add a space to the output string, but letters are converted to uppercase for consistency, converted into an ASCII value, then subjected to a modulus by 65 operation to put it in the 0-25 alphabet range. The numeric value of the key is added and another modulus operation by 26 is done to cover for wrapping from Z to A. 65 is added back to the ASCII value then converted back into a character. This character is then concatenated into the output string.  
   This code functions as intended, as modulus operations account for circular wrapping, and the use of ASCII values simplifies indexing the letters.  
   Decrypt: The decrypt algorithm functions essentially the same as the encryption, however the index shift has slight adjustments to the operations. The obvious change is that the ASCII value is subtracted by the index rather than added. The other change is that the ASCII value is added by 26 after the modulus by 65 operation to accommodate for circular shifting in the A->Z direction (can’t be hitting negative values). The modulus by 26 operation clears this addition by 26 if no circular shift occurs.  
   Because the decryption code has the same foundation as the encryption, it works as expected because it adequately accounts for circular shifting in the opposite direction.  
   StatAttack: For this algorithm, an empty 26 long array is generated to represent the amount of occurring instances of each letter in the string input. A For loop iterates though the string input and checks for letters and their ASCII index, so that the appropriate spot in the array can be incremented by 1. A unique length for characters is also incremented every time a character is passed in. Once the index array done reading in characters, the algorithm transitions to a separate for loop using the index array. A new array is generated every run of the loop till all values of i are accounted for. The loop array then reads in the values of the index array but circularly shifted by i places in the array. A double variable for correlation then adds up all correlations from the unigram multiplied by the ratio of occurrences of a letter divided by the total amount of characters. If the resulting correlation is higher than the input threshold, the code will print out the value of I, the value of the correlation, and call the decryption algorithm to print out the decrypted text using that value of i.  
   The code works as intended simply using the array to track the instances of letters makes it easy to manipulate by shifting for values of i, which was the most difficult part of the algorithm.
3. Conclusions  
   One of the first things realized during the assignment was how tricky circular shifting would be with letters, due to the conversion to ASCII and a lack of built in shift functionality to Matlab. Generally, a longer and more varied word (lower character frequencies) is more difficult to attack. Plaintext length very much matters as well as the ratio of character frequency. Though the code works as intended, edits could made to the ASCII value operations to simplify them, such as making index shifts on the base ASCII values and adjusting appropriately only when circular shifting occurs (by checking with if statements). I would still use the implemented algorithm for letter shifting, but the above concept is an equally viable option.
4. Appendix

Encrypt:  
function EncryptCaesar = Encrypt(stringIn,key)

%UNTITLED Summary of this function goes here

% Detailed explanation goes here

modify = char(stringIn);

EncryptCaesar = "";

length = strlength(stringIn)

for i = 1:length

check = modify(i);

if ( isletter(check))

check = upper(check);

checkN = double(check);

checkN = mod(checkN, 65);

checkN = checkN + key;

checkN = mod(checkN, 26);

checkN = checkN + 65;

check = char(checkN);

EncryptCaesar = strcat(EncryptCaesar, check);

elseif (isspace(check))

EncryptCaesar = strcat(EncryptCaesar, " ");

else

end

end

disp(EncryptCaesar)

end

Decrypt:

function EncryptCaesar = Decrypt(stringIn,key)

%UNTITLED Summary of this function goes here

% Detailed explanation goes here

modify = char(stringIn);

EncryptCaesar = "";

length = strlength(stringIn);

for i = 1:length

check = modify(i);

if ( isletter(check))

check = upper(check);

checkN = double(check);

checkN = mod(checkN, 65);

checkN = checkN + 26;

checkN = checkN - key;

checkN = mod(checkN, 26);

checkN = checkN + 65;

check = char(checkN);

EncryptCaesar = strcat(EncryptCaesar, check);

elseif (isspace(check))

EncryptCaesar = strcat(EncryptCaesar, " ");

else

end

end

disp(EncryptCaesar)

end

StatAttack:

function integer = Stat(stringIn,threshold)

%UNTITLED2 Summary of this function goes here

% Detailed explanation goes here

array = zeros(1,26);

modify = char(stringIn);

length = strlength(stringIn)

charlength = 0;

for i = 1:length

check = modify(i);

if (isletter(check))

check = upper(check);

checkN = double(check);

checkN = mod(checkN, 65) + 1;

array(checkN) = array(checkN) + 1;

charlength = charlength +1;

else

end

end

for i = 0:25

correlation = 0.0000;

newarray = zeros(1,26);

for s = 1:26

x = s + 25;

x = x - i;

x = mod(x,26) + 1;

newarray(x) = array(s);

end

correlation = correlation + (0.080\*(newarray(1)/charlength));

correlation = correlation + (0.015\*(newarray(2)/charlength));

correlation = correlation + (0.030\*(newarray(3)/charlength));

correlation = correlation + (0.040\*(newarray(4)/charlength));

correlation = correlation + (0.130\*(newarray(5)/charlength));

correlation = correlation + (0.020\*(newarray(6)/charlength));

correlation = correlation + (0.015\*(newarray(7)/charlength));

correlation = correlation + (0.060\*(newarray(8)/charlength));

correlation = correlation + (0.065\*(newarray(9)/charlength));

correlation = correlation + (0.005\*(newarray(10)/charlength));

correlation = correlation + (0.005\*(newarray(11)/charlength));

correlation = correlation + (0.035\*(newarray(12)/charlength));

correlation = correlation + (0.030\*(newarray(13)/charlength));

correlation = correlation + (0.070\*(newarray(14)/charlength));

correlation = correlation + (0.080\*(newarray(15)/charlength));

correlation = correlation + (0.020\*(newarray(16)/charlength));

correlation = correlation + (0.002\*(newarray(17)/charlength));

correlation = correlation + (0.065\*(newarray(18)/charlength));

correlation = correlation + (0.060\*(newarray(19)/charlength));

correlation = correlation + (0.090\*(newarray(20)/charlength));

correlation = correlation + (0.030\*(newarray(21)/charlength));

correlation = correlation + (0.010\*(newarray(22)/charlength));

correlation = correlation + (0.015\*(newarray(23)/charlength));

correlation = correlation + (0.005\*(newarray(24)/charlength));

correlation = correlation + (0.020\*(newarray(25)/charlength));

correlation = correlation + (0.002\*(newarray(26)/charlength));

if (correlation > threshold)

output = "";

output = strcat(output, num2str(i) + ", ");

output = strcat(output, num2str(correlation) + ", ");

stringCode = Decrypt(stringIn, i);

output = strcat(output, stringCode);

disp(output);

end

end

integer = 1;

end

Test Code:

Encrypt(“Hello World.”, 4)

Decrypt(“LIPPS ASVPH”, 4)

Stat (“Hello World.”, 0.05)

Stat (“LIPPS ASPVH”, 0.05)

Stat (“KHOOR ZRUOG”, 0.05)







