SECRET CODES WITH CAESAR CIPHERS
MATH 355A - NUMERICAL ANALYSIS

ALEX ANDOINIAN AND CAMDEN BOCK

1. Interactive Lecture

1.1. Objective.

Outcomes:

- Students will be able to add two-digit numbers, modulo 26.
- Students will be able to encode and and decode strings of characters using a Caesar Cipher and key.
- Students will be able to list at least two modern or historical applications of cryptography.

Purpose:

- Students will be exposed to applications of mathematics, computer science and cryptography.
- Students will appreciate the uses of mathematics and computer science outside the traditional classroom.
- Students will investigate interests in a career or future studies in mathematics and computer science.

Concepts to be learned:

- Students will understand how to use a Cryptogrpahic wheel to encode and decode Caesar ciphers.
- Students will understand the division algorithm, and how to find a remainder.
- Students will understand how to apply the division algorithm to modular arithmetic.
- Students will understand how to apply modular arithmetic to endcoding and decoding characters in a Caesar cipher.

Curriculum Reference:

- Common Core State Standard 8.FA1-3: Define, evaluate, and compare functions
- Common Core State Standard 8.FB4-5: Use functions to model relationships between quanities
- Common Core State Standard 8.EEC-7A-B: Solve Linear Equations with One Variable

1.2. Preparation.

Date: February 5, 2015.

Teacher Preparation: Teachers should be familiar with Caesar ciphers using cryptographic wheels, and familiar with modular addition. Teachers should also be aware of the modern applications of cryptography, and it's historical uses.

Student Preparation: Students should assemble the cryptographic array, and have been introduced to modular addition. This introduction might be related to (mod 12) and (mod 60) in telling time.

Materials:

- Cryptographic Wheel
- $2 \times n$ array handouts

1.3. **Procedure.**

Introduction:
The lesson will be introduced with an general explanation of the purpose of cryptography, as the science of encoding and decoding information to allow for secure communication. A number of modern applications will be presented, including electronic currency and network communications.

Content:

Introduction of the Terminology of Cryptography
Terminology will be introduced with definitions, and the meaning of the new terms will be repeated as they are used in context.

- Plain text: text as printed by a word-processor
- Cipher text: encrypted plain-text from an encryption algorithm
- Cipher: an algorithm that encodes information
- Algorithm: a procedure executable by a human or computer
- Key: an index that allows an encoded message to be decoded

Overview of the History of Cryptography, SecurityBlog at RedHat

- Carving in Egyptian tomb (1900BC)
- Juilus Caesar military application (100BC)
- Vigenere (introduction of encryption key) (1500sAD)
- Electro-mechanical encryption (1800s AD)
- Engima machine, Germany (WWII)
- IBM "crypto group" produces "Lucifer" cipher (1970s)

Caesar Ciphers

- Introduction to the Caesar Cipher using Cryptographic wheels
- Introduction to modular arithmetic
- Encryption via Caesar Cipher with modular arithmetic
- Allow students to pass encrypted message/note to friend
- Discussion about possible methods of decryption/amount of security Caesar cipher provides

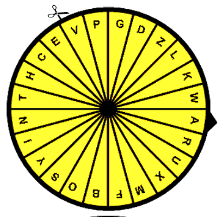
Conclusion:
After explicit instruction, and working through a number of examples, students will have an opportunity to encode, send, and decode messages to each other.

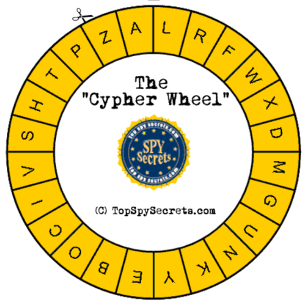
Extensions:

- What are some ways we could make the Caesar cipher more secure?
- Are there ways to 'break' or decrypt a Caesar cipher? What if we had more information on the frequency of letters?
- What happens if we use $\pmod{20}$ instead of $\pmod{26}$ for a Caesar cipher?
- Can you think of another way to encrypt messages?
- How can we use modular arithmetic for Vigenère ciphers?

Differentiation:
New terminology will be defined before the lesson aurally and textually, and be accessible throughout the lesson. The definitions of new terminology will be incorporated in to the explanation of new concepts. During the Lab section, pseudo-code will be on the board, available as a handout, and discussed aurally. Instructors will be use a projected workstation to demonstrate some of the essential steps in creating the program, and other instructors will be available for one-on-one or small group instrution at students' workstations.

Attachments:
History of Cryptography:
`https://securityblog.redhat.com/2013/08/14/a-brief-history-of-cryptography/`
Worksheets and Cryptographic Wheels:
`http://www.math.uic.edu/CryptoClubProject/CCpacket.pdf`





2. Lab Activity

2.1. **Objective.**

Outcomes:
Students will be able to perform the following skills:

- Navigate the MATLAB environment (e.g. open/save/run scripts and functions, make commands in the command window, switch current directory etc)
- Preform basic mathematical operations (use MATLAB as a handheld calculator)
- Store, retrieve and manipulate variables
- Prompting user for input and storing for use
- Ensuring that users input satisfies certain conditions: if statements
- Performing operations on input: for loops
- Displaying final results

Purpose: In this lab activity, students will first receive a tutorial on the basics of MATLAB programming. Next, students will use these computer programming skills to aid in the implementation of a MATLAB script, Caesar.m, whose purpose is to encode messages using a Caesar cipher. Implementing the Caesar cipher in MATLAB should reinforce the lessons taught in part 1, and students to begin thinking about the problem algorithmically (as a series of logical steps), and they must understand each step to ensure a successful program.

Concepts to be learned:

- Students will understand how to manipulate basic data structures in MATLAB
- Students will understand how to input data in MATLAB
- Students will understand how to print the output of a program, in context.
- Students will understand how to implement an algorithm based on a computational strategy.

2.2. **Preparation.**

Teacher Preparation: The instructor should be familiar with the attached implementation of the Caesar Cipher, and with basic MATLAB commands. The attached implementation includes boiler-plate code with which they can use to complete the program, but it will also incorporate many stylistic aspects of code that are common practices including commenting, proper indentation, and sectioning. The instructor should also be familiar with using MATLAB to call the "helper functions" that convert character strings to arrays of ASCII values. Instructors should be familiar with the basics of ASCII values and their equivalent characters.

2.3. **Procedure.**

The teacher will use the Pseudo-code, as a handout, and walk-through the Caesar Cipher implementation with students working in small groups at workstations, and the instructor's workstation projected.

Pseudo-code:

Get necessary input from user and store in variables

- Prompt user for plain text to be encrypted
- Prompt user for rotational key
- *Hint: consider the function "input(This is my secret message!)"*

Prepare for Encryption

- Convert plain text (characters) to numbers using ASCII coding scheme
- Initialize the appropriately sized array to store cipher text

Process Plain Text

```
    for every letter in the plain text
        if the letter is uppercase
            shift letter k positions

        or if the letter is lowercase
            shift letter k positions

        if not upper or lowercase, then must be number or symbol
            keep the character the same
    end
```

Hints: When shifting letters, think about using the modulo operator, which can be called in MATLAB using mod(x,y).

```
%Within a variable, say x, you can access the ith element in x with ...
    the command
EDU >> x(i)
%For example, if you type
EDU >> x =  Hello
%Then typing
EDU >> x(2)
%Will produce
ans =
    e
```

Display the encrypted message.

- Convert numbers back to characters.
- *Hint: consider the function "disp('Guvf vf zl frperg zrffntr!')"*

MATLAB Code:

Caesar Cypher

```
%% *** Caesar Cipher ***
% by Alex Andonian
%%% Implements a ROT K Caesar Cipher
%%% Accepts a string (plain_text) and rotational key (k) and ...
    displays an
%%% encrypted message whereby each letter in plaintext is rotated by
%%% k positions in the alphabet, wrapping around from 'Z' to 'A' as
%%% needed.
%%% More formally, for some plaintext p, where p_i is a character in...
     the
%%% plaintext and k is the rotational key, then the corresponding
%%% character c_i in the cipher text is computed with:
%
%   c_i = (p_i + k) % 26

%% Get Input
% Get plaintext from user
plain_text = input('Please enter message to be encrypted:\n','s');
% Get rotational key from user
k = input('Please enter a rotational key \n');
% Ensure proper input
if (mod(k,1) ≠ 0)
    warning('Rotational key must be an integer; However, one was not...
         provided. Input was rounded to nearest integer.');
end
%% Process Plaintext
% Convert char to double using ASCII coding scheme
plain_text = double(plain_text);
% Intialize array to store ciphertext
cipher_text = zeros(1,length(plain_text));
% Iterate over every letter in the plain text
for i = 1:length(plain_text)
    % process uppercase letters
    if (plain_text(i) ≥ double('A') && plain_text(i) ≤ double('Z'))
       cipher_text(i) = mod(plain_text(i) - double('A') + k, 26)+ ...
           double('A');
    % process lowercase letters
    elseif (plain_text(i) ≥ double('a') && plain_text(i) ≤ double('z'))
        cipher_text(i) = mod(plain_text(i) - double('a') + k, 26)+ ...
            double('a');
    % leave symbols and numbers untouched
    else
        cipher_text(i) = plain_text(i);
    end
end
%% Display Encrypted Message

% Convert double to characters using ASCII coding scheme
cipher_text = char(cipher_text);
disp('Your encrypted message:');
disp(cipher_text);
```

Provided Function to Convert Character Arrays to Arrays of Integers

```
function [num] = charToNum(x)
%charToNum Converts an array of characters to an array of numbers where
%each letter is converted to its corresponding position in the ...
    alphabet starting from 0.
% e.g. charToNum('A') = 0
num = double(x);
for i = 1:length(num)
    % process uppercase letters
    if (num(i) ≥ double('A') && num(i) ≤ double('Z'))
       num(i) = num(i) - double('A');
    % process lowercase letters
    elseif (num(i) ≥ double('a') && num(i) ≤ double('a'))
        num(i) = num(i) - double('a');
    end
end
```