New College of Florida – Spring 2024

CSCI 4525 – Assignment 1

Due by: February 22, 2024, 11:55PM

Instructions:

1. Answer all questions, and show all your work. Marks cannot be awarded for what is not submitted. Do not look up your answers online, remember your work must be solely yours.
2. Ensure your work is legible. Your written work may be given by hand or typed up, regardless the written work must be submitted as a paper copy. It is advised if you are writing your answers by hand to print out program output as opposed to handwriting those parts of the written problems.
3. Have a title page including your name (underlining your family name), your student e-mail/prefix, the course number, your instructor's name, assignment number, and the date completed. Your assignment submission must not have any loose pages, i.e. please staple your assignment (no staples will be provided). Number the pages of your assignment, and make sure you number problems as they correspond in the assignment.
4. Submit your assignment to your instructor, ask the secretary in-person in the Natural Science office to place it in the instructor's mailbox, or carefully slide it under the instructor's office door (Heiser 155). I will not accept assignments submitted **after** I arrived to my office the next morning (8:30AM or later) following the deadline. To guarantee the instructor receives your assignment, you may give it to the instructor the day of class.
5. Submit any code you write to the Canvas item for this Assignment. Your programs can be written in Java, C/C++, or Python. You need to state in your assignment submission which language code was written in. Your written work must refer to me to particular files when referring to code, so I understand where to look. It will be assumed that all files can be placed in the same directory, should they be executed. Whenever possible, name your program files by problem number.

Problems (30 marks)

Beside each problem number is if the problem has a written, programming, or written/programming component.

1. [5 marks] **What a Shift!?** [Written/Programming] Ignoring the blank spaces, decrypt the following string

```
ZWQPF LQUFE FKQNF IBQFE QZDGF IKREK QGIFS CVDJQ QZKQZ JQEFK QCZBV CPQKY
RKQPF LQNZC CQUFQ ZDGFI KREKQ NFIBQ QIZTY RIUQY RDDZE X
```

which was encrypted using a shift cipher.

Determine using an exhaustive-key search the original English message from plaintext, and the key k used to encrypt its corresponding plaintext.

a) [Programming] Implement the decryption algorithm for the shift cipher, and use it to help you. You do not need to submit in your written work your code, you will submit your code on Canvas.

b) [Written] Describe what you did in your written work, and explain briefly how you obtained the desired plaintext. Include program output in your written work and explain how you used it.

c) [Written] Finally, present the key k you found and tidy up the deciphered (correct) message. Note that the secret message is a quotation of a pioneering computer scientist.

2. [8 marks] **The Divisor Greatest of All.** Consider the classic algorithm for computing the greatest common divisor of two positive integers a and b, called the *Euclidean Division Algorithm*. Please review the attached short reading from [S] on the Euclidean Division Algorithm and Extended Euclidean Division Algorithm.

a) [2 marks] [Written] For $a = 420$ and $b = 17$, give the values of the quotients and remainders as a result of applying the variant of the Euclidean Division Algorithm given in the reading. Show your work similar to on page 188 of [S], where you present the q_i's and r_i's clearly [or at the end], indicating the final result r_m. Start with $r_0 = a$ and $r_1 = b$.

b) [3 marks] [Programming/Written] Implement the variant of the Euclidean Division Algorithm given as Algorithm 6.1. Have your implementation after determining the greatest common divisors print $(q_0, q_1, q_2, \ldots, q_m)$ and then $(r_0, r_1, \ldots, r_m, r_{m+1})$, then at the end of the algorithm's implement return r_m. Please submit your program on Canvas. In your written work, present the program output to check your work from part (a).

> As a tip, if you are using Lists/ArrayLists/Vectors (lists), create two lists, one a list for each q_i and another list for each r_i, and add q_m and r_m into the list when it is to be introduced each iteration of the loop appropriately. You are allowed to make a value $q_0 = 0$ to put at the start of the list for the q_i's to make the algorithm as presented easier to implement.

c) [3 marks] [Written] The variant of the Euclidean Division Algorithm implemented in the previous part stores all the quotients and remainders during each iteration of the main while loop. Suppose we only wished to compute r_m [not printing all the q_i's and r_i's], explain at least one way briefly we can reduce the memory usage of Algorithm 6.1. After, present a more space-efficient version of the algorithm as pseudocode based on your idea.

> Remember that r_{m+1} is not the greatest common divisor, value r_m is!

3. [3 marks] **There Exist Inverses!** [Written/Programming] Consider $\mathbb{Z}_{36} = \{0, 1, 2, \ldots, 35\}$.

a) [2 marks] [Programming/Written] List all elements in $\mathbb{Z}_{36}$ that have a multiplicative inverse, and state how many multiplicative inverses there are in total. Use your program from the previous problem to compute this. You can either implement the space-improved version of the Euclidean Division Algorithm or omit the print statements giving the lists, for ease of reading. Submit your program on Canvas. In your written work, provide your program's output.

> Do note that 0 never has a multiplicative inverse, because there is no integer that you can multiply with it modulo m that equals 1.

b) [1 mark] [Written] Check that you have the correct number of multiplicative inverses using the theorem from class involving the Euler ϕ-function. Show your work by hand computing this value.

4. [4 marks] [Written] Consider the shift cipher over $\mathbb{Z}_{24}$. Present all possible keys $k \in \mathcal{K} = \mathbb{Z}_{24}$ that have the following property:

$$\text{For key } k \text{ and every } x \in \mathbb{Z}_{24},$$

$$e_k(x) = d_k(x).$$

Show your reasoning for how you determined all such keys (and no others).

Note: In plain language, this property guarantees that any plaintext symbol is the same as its ciphertext symbol.

> **Hint:** Use definitions of the encryption/decryption methods $e_k(x)$ and $d_k(y)$ when $x = y$, and perhaps properties of congruences.

5. [4 marks] **Lots of Affinity.** [Written] For this problem we consider the affine cipher with alphabetical letters over $\mathbb{Z}_{26}$.

a) [2 marks] [Written] By hand, encrypt the following plaintext with key $K = (3, 5)$. Please state the resulting ciphertext at the end with alphabetical symbols.

```
encrypt
```

b) [2 marks] [Written] Decrypt the ciphertext obtained in (a) by hand as part of this work (remember, you have a for the key). Begin with the alphabetical symbols of the ciphertext. Show your work.

6. [6 marks] **Brute-Force and Affine.** [Programming/Written] Decrypt the following ciphertext (over $\mathbb{Z}_{26}$), which was encrypted with an affine cipher, using an exhaustive-key search. Present the original message cleaned up and key $K = (a, b)$.

```
EILYT AELJD ILHGC JLNQJ LUGTT AJLUA TJVAF LKJZL EILUA TJVAF LJDIL PQJQV
ILNQJ LUGTT AJLYT AELKJ Z
```

Using a similar format as problem (1), present your work. Do note that unlike in problem (1), you are only responsible for presenting the program output for all b-values for the a with the key $K = (a, b)$ you found, due to the larger key space; however, your program should carry out computations to assist you in your cryptographic analysis. More discussion is provided on the next page for decryption algorithm's implementation.

To accomplish your attack, implement the decryption algorithm of the affine cipher. To do this, implement the below algorithm from p. 192 of [S], which is a variant of the Extended Euclidean Division Algorithm for computing the multiplicative inverse. Note you are welcome to use any implementation of the Euclidean Divison Algorithm to check if a key is valid first or simply return -1 from your implementation of the below algorithm whenever the multiplicative inverse does not exist. Submit your program on Canvas.

Algorithm 6.3: MULTIPLICATIVE INVERSE(a, b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$
do $\begin{cases} temp \leftarrow (t_0 - qt) \bmod a \\ t_0 \leftarrow t \\ t \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$
if $b_0 \neq 1$
then b has no inverse modulo a
else return (t)

Two technical notes about this pseudocode:

- The above algorithm computes $b^{-1} \bmod a$ presuming it exists. So, be mindful of how you make calls to this method. You are welcome to re-arrange the arguments (or swap a and b's roles in the above pseudocode) to make it easier to use for your implementation, e.g. make it compute a^{-1} instead.
- 0^{-1} never exists, so it is not accounted for in [S]'s pseudocode.

Do not submit work from the following suggestion: Consider check for yourself the multiplicative inversed computed against the list of multiplicative inverses of integers in $\mathbb{Z}_{26}$ relatively prime to 26 on page 22 of our course notes.