COMP 7402 (FLEX) Cryptology Assignment #1

Your submission should be in a **package** `<FirstName_LastName_StudentID.zip>` and include:

- A **PDF** document report containing answers to all questions including typed **and/or** hand-written answers to the descriptive questions, tools that you used, any supporting data, and references.
- Ensure your report begins with a table of contents for easy navigation.
- Source **code** files.
- **Screenshots** of the executed code.
- Please note that during the lab, I may ask you questions about the details of your implementation to verify that you completed the assignment yourself. If you are unable to satisfactorily explain your answers, you will not receive any marks for the assignment.

To be completed by **Jan 23, 2025**. This is an **individual** assignment. Late submissions will **not be accepted.** Total mark is **100**.

TASK 1: NUMBER THEORY PRACTICE QUESTIONS AND PROBLEMS [50 MARKS]

1. Given a = -12 and n = 7. Derive an equation that relates both integers using the division algorithm **[5]**

2. Use the Euclidean algorithm to determine GCD (24140, 16762) **[5]**

3. Find two integers x and x′ that satisfies the equation $5x \equiv 4 \pmod{3}$ **[5]**

   - Note that $5x - 4 = 3k$ (the properties of congruencies)

4. Use the extend Euclidean algorithm to find the modular multiplicative inverse of 1234 mod 4321**[10]**

5. Using Fermat's theorem find 3^{201} mod 11 **[5]**

6. Use Euler's Theorem to find a number a between 0 and 9 such that a is congruent to 7^{1000} mod 10 **[5]**

7. The Miller-Rabin test can determine if a number is not prime but cannot determine if a number is prime. How can such an algorithm be used to test for primality? **[5]**

8. Given 2 as a primitive root of 11 construct a table of discrete logarithms **[10]**

i	2^i mod 11	a	Log $_{2,11}$ a
1			
2			
3			
…	…	…	…
10			

TASK 2: EXTENDED EUCLIDEAN ALGORITHM IMPLEMENTATION [25 MARKS]

Using **any** programming language of your choice **(preferably python or java)** implement the **Extended Euclidean algorithm [10]**

Specifications

Your function should take two inputs:

1) An integer **a**, which is the modulus
2) A non-negative integer **b** that is less than **a**.

and output three values:

1) **gcd(a,b)**
2) Integer **x** and
3) Integer **y**, such that **ax + by = gcd(a,b)**

Test 1 [5 Marks]

1) Run your program with ***a = 43 b = 17***
2) What are your outputs?
3) What is the modular multiplicative inverse of ***17 mod 43***? Note that the modular multiplicative inverse has to be non-negative and less than 43.

Test 2 [5 Marks]

1) Run your program with ***a = 400 b = 10***
2) What are your outputs?
3) What is the modular multiplicative inverse of ***10 mod 400***? Be mindful of the GCD value to answer this question.

In your task submission, include the following

A **README** file, which should explain how to run the code with sample input and output. If you are unfamiliar with READMEs, you can find an introduction here:

https://www.makeareadme.com/
and here
https://medium.com/@meakaakka/a-beginners-guide-to-writing-a-kickass-readme-7ac01da88ab3

Note that the README file you submit for this project need not be complex, it only needs to at least explain how to compile the code and run the code with examples. **[5]**

TASK 3: IMPLEMENTING THE PLAYFAIR CIPHER [25 MARKS]

Using **any** programming language of your choice **(preferably python or java)** implement the Playfair cipher. We know that Playfair cipher uses digraph substitution to provide more complexity compared to monoalphabetic ciphers (such as Caesar that we discussed). **[10]**

Example: Playfair Cipher - University of California, Berkeley

10 marks BONUS if you use MATLAB: Downloading MATLAB for Student Home Use for BCIT Students

Specifications

1) Write a code that takes a plaintext input from the user.
2) Your program should create a 5x5 matrix (key table) based on a keyword
3) Use the key table to encrypt the plaintext by the rules of the Playfair
4) Pay no attention to punctuation or to spaces between words.
5) Note that there might be small differences in the implementations of the Playfair cipher in the literature

Rules for the Playfair Cipher:

- Remove any duplicate letters from the keyword.
- Fill the 5x5 matrix with the letters of the keyword and then the remaining letters of the alphabet (excluding 'J', which should be treated as 'I').
- Divide the plaintext into digraphs (pairs of two letters). If a digraph consists of the same letter (like "LL"), insert an 'X' between them ("LXL").
- If the plaintext has an odd number of characters, append an 'X' to make it even. For each digraph:
  - If both letters are in the same row, replace them with the letters to their immediate right (wrapping around if necessary).
  - If both letters are in the same column, replace them with the letters immediately below them (wrapping around if necessary).
  - If the letters form a rectangle, replace them with the letters on the same row respectively but at the other pair of corners.
  - You can find these rules in page 31 of the lecture slides

Input: A keyword <string> and plaintext <string> from the user | **Output:** The encrypted ciphertext.

Test Cases: Include ***three*** test cases (preferably edge cases) to verify your implementation. Use the keyword **MONARCHY** (we discussed ***iiі*** in the lecture, try this one as a test case, you can also try words like balloon, etc., to show the edge cases) **[10]**

Here are some tools you can verify your answers: https://planetcalc.com/7751/ and https://www.boxentriq.com/code-breaking/playfair-cipher

In your task submission, include the following

A **README** file, which should explain how to run the code with sample input and output. **[5]**