ECE406 **Algorithm Design & Analysis** **Winter 2025**

Assignment #2
Due Fri Jan 31 by 11:59pm
See Appendix for submission instructions

Exercise 1 (Programming Question: Prime Numbers and RSA) Download the python script called `ece406w24-A2.py` from LEARN. You will also need `modexp` and `extended-Euclid` from assignment 1. Complete the following questions and submit your completed python file on **LEARN**.

On **Crowdmark**, submit a screencap of your pseudocode for parts (i) and (ii), and submit the values you obtained for parts (iii) to (vi).

(i) Write a function `primality(N)` that tests if a number N is prime using Fermat's little Theorem. Your algorithm should test N using 10 values of a, each randomly drawn from $\{1, \ldots, N-1\}$. If $a^{N-1} \mod N = 1$ for all ten values of a, then the function returns `True`, and otherwise it returns `False`.

(ii) Write a function `prime_generator(N)` which generates a prime number $\leq N$ using the function `primality`. Make sure your function eventually terminates.

(iii) Generate two 7-digit prime numbers p and q using your prime number generator. Using the encoding exponent $e = 5$, check if e and $(p-1)(q-1)$ are relatively prime. If they are not, then re-generate p and q until you find ones such that e and $(p-1)(q-1)$ are relatively prime.

(iv) Find the value of d that should be used for the private key.

(v) What is the encryption of the message $x = 2148321$ (x is in decimal)?

(vi) Verify that your decoding exponent d works correctly by decoding your message.

NOTE: This is a fairly open-ended programming question, so we do not provide unit tests. Instead, write `print` statements into the `main` function to show that your code completes the instructions for parts iii–vi.

Exercise 2 (Recurrence Relations) Solve the following recurrence relations to determine $T(n)$ (in big-O notation):

(i) $T(n) = 3T(n/2) + O(1)$

(ii) $T(n) = 4T(n/5) + O(n)$

(iii) $T(n) = 4T(n/8) + O(n^2)$

(iv) $T(n) = T(n-1) + n$ (Note, this does not fit the form of the Master Theorem. However, you should be able to solve it exactly by assuming n is even, and repeatedly substituting in the definition until you get $T(n)$ as a function of $T(0) = 0$).

Exercise 3 (Divide-And-Conquer) Given a sorted array of distinct integers $A[1 \ldots n]$, give a divide-and-conquer algorithm that runs in $O(\log n)$ time and answers the following question: "Is there an index i for which $A[i] = i$?" Answer the question by writing the pseudo-code for your algorithm and characterizing its run-time $T(n)$.

Exercise 4 (Divide-And-Conquer) Consider the following function

```
function f(n)
   if n > 1:
       for i = 1 ... n:
           print("yes")
       end
       f(n/3)
       f(n/3)
       f(n/3)
   end
```

Assuming n is a power of 3, then how many times does $f(n)$ print the word "yes." Give your answer in $O(\cdot)$ by writing a recurrence and solving it (your answer should be the smallest upper bound possible).

Appendix

How do I submit this assignment? To submit the assignment, please complete the following steps.

(i) Upload your typed or handwritten work for *each* question using the ECE406 **Crowdmark** site. This includes your answers to the programming questions, where you should submit screenshots of the Python functions you have implemented.

(ii) Upload your source code for each programming question to the **A2 Dropbox submission folder** in LEARN. The starter code for each question specifies the filename convention you should follow.