Problem Set 2

Undetectable errors are infinite in variety,
in contrast to detectable errors,
which by definition are of limited flavors.
— *Murphy's Law (CS variant)*

- This problem set is due **at 9:00am** on **February 10, 2025** .
- This problem set comprises 3 problems.
- Submit early. Do **not** wait till the last moment.
- Each solution should start on a new page.
- A sample file will be provided on piazza with blank functions for you to fill. Use it.
- We will give full credit **only** for correct solutions that are described clearly and convincingly.

Problem 2-1. More and More History [40 points]

Around 2300 years ago, Euclid published his magnum opus, *Elements*, containing one of the earliest and simplest algorithms still in common use. His method described a simple, step by step way of calculating the greatest common divisor of two natural numbers. You have almost certainly learned this method in school. Write OCaml code for each of these.

(a) [10 points] The original version was based on subtraction: given two numbers a and b, you would update $a = a - b$ (if a was bigger than b) or $b = b - a$ otherwise, until a was equal to b. This last value would be your solution.

(b) [10 points] An updated version used remainders, finding the GCD of a and b by instead solving for the GCD of b and the remainder of a divided by b.

(c) [20 points] In the 1800s, Erienne Bezout wrote what is known today as Bezout's identity: Let a and b be integers with greatest common divisor d. Then there exist integers x and y such that $ax + by = d$. Moreover, the integers of the form $az + bt$ are exactly the multiples of d.

The values of x and y above are incredibly important for a variety of applications (especially in cryptography, e.g., for finding modular inverses). These are calcuated using the *Extended Euclidean Algorithm*, which, given a and b, returns x and y such that $ax + by = GCD(a, b)$.

A perfectly good description for how you can do this is available on the wikipedia page about the topic:
`https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm`

Problem 2-2. The Code that Broke OCaml's Back [15 points]

The following OCaml functions contain bugs. Identify and fix them for 5 points each.

```
(* factorial 5 should return 120 *)
let rec factorial (n : int) : int =
  if n = 0 then 0
  else n * factorial (n - 1)

(* fibonacci 5 should return 5 *)
let rec fibonacci (n : int) : int =
  if n = 0 then 0
  else if n = 1 then 1
  else fibonacci (n - 1) + fibonacci (n - 2) + 1

(* maybe your GCD code wasn't working! *)
let rec gcd a b =
  if a = b then a
  else if a > b then gcd a - b b
  else gcd a b - a
```

Problem 2-3. Write a function to write a function to write a ... [45 points]

Write code for the following. For each of these, you should also write a short note on what it does: try to make your note understandable by an average class-10 student. This note should be there as a comment (in one piece) at the beginning of your code. Your code is worth 10 points, and your note is worth 5.

(a) [15 points] A naive way to compute x^n is to multiply by x repeatedly, accumulating the answer and returning it. There is, however, a faster way, owing to the fact that any natural number can be written as a sum of powers of 2.

If n is even, x^n can be written as $(x \times x)^{n/2}$.
If n is odd, x^n can be written as $x \times (x \times x)^{(n-1)/2}$.

Use these facts to write a quick way to compute x^n.

(b) [15 points] The Ackermann function is a fascinating function which grows very rapidly.

The function is defined as:
$A(0, n) = n + 1$
$A(m + 1, 0) = A(m, 1)$
$A(m + 1, n + 1) = A(m, A(m + 1, n))$

(c) [15 points] Starting with an arbitrary positive integer, do the following: if the number is even, divide it by two; otherwise, triple it and add one.

The Collatz Conjecture states that doing this repeatedly will eventually produce 1, no matter what value one starts with. Count the number of Collatz steps taken to reach 1, starting with a number x.