We may grade a **subset of the assigned questions**, to be determined after the deadline, so that we can provide better feedback on the graded questions.
Unless otherwise stated, each question requires sufficient justification to convince the reader of the correctness of your answer.
For bonus questions, we will not provide any insight during office hours or Piazza, and we do not guarantee anything about the difficulty of these questions.
We strongly encourage you to typeset your solutions in LaTeX.
If you collaborated with someone, you must state their name(s). You must write your own solution for all problems and may not look at any other student's write-up.

0. If applicable, state the name(s) and unique name(s) of your collaborator(s).

Solution:

1. Suppose that *input()* is a function that returns a user-specified positive integer. For each of the following programs, do the following:

   - State whether or not the program necessarily halts for *all* possible sequences of user-provided (positive integer) inputs.
   - Justify your answer, either by providing a proof of termination via a potential-function argument, or by describing a sequence of valid inputs that causes the program to run forever.

   Note: z is input repeatedly, which means it can be different each time it is read.

(a)

```
1:  x ← input()
2:  y ← input()
3:  while x > 0 and y > 0 do
4:      z ← input()
5:      if z is even then
6:          x ← x + 4
7:          y ← y − 5
8:      else
9:          y ← y + 5
10:         x ← x − 4
```

Solution:

(b)

```
1: x ← input()
2: y ← input()
3: while x > 0 and y > 0 do
4:     z ← input()
5:     if z is even then
6:         x ← x − 3
7:         y ← y + 5
8:     else
9:         y ← y − 2
```

Solution:

2. Let $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ be a solution to $\varphi^2 = \varphi + 1$. Show that the number of iterations (not including the base case) made by Euclid's algorithm on (x, y) is at most $\log_\varphi(x + y)$. Note that this is a slight improvement over the bound of $\log_{3/2}(x + y)$ that was shown in class.

   Hint: Change the potential function from class to $s(x, y) = cx + y$ for a suitable constant c. Try using $c = \varphi$. Consider the crucial point that $\varphi + 1 = \varphi^2$ and/or other similar relations satisfied by φ.

   Solution:

3. Alice is playing a `FactorFinding` game with herself. The integer factorization is not exciting enough so she decides to consider another number system.

   Alice thinks about the "complex integers", i.e. $a + bi$ where a, b are integers and $i^2 = -1$. For example,

   - addition: $(1 + 2i) + (3 + 4i) = 4 + 6i$;
   - multiplication: $(1 + 2i)(3 + 4i) = 3 - 8 + (6 + 4)i = -5 + 10i$

   She then plays the game as follows. Alice starts with an arbitrary "complex integer" $a_1 + b_1 i$ and $k = 1$. Alice tries to find $a_{k+1} + b_{k+1} i$ that divides $a_k + b_k i$ which means there exist two integers c and d such that $(a_{k+1} + b_{k+1} i)(c + di) = a_k + b_k i$. If $|c| + |d| \leq 1$, then the game terminates. Otherwise, she increments k and repeats this step.

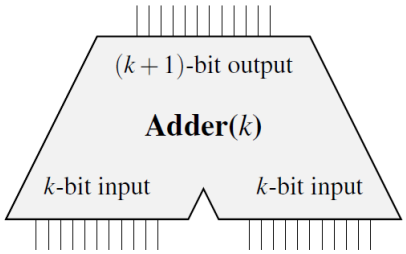
   Question: Can Alice continue the game forever? If yes, give an example infinite run. If not, prove it by the potential-function method.

   Hint: Try the potential function $\Phi(a + bi) = a^2 + b^2$.

   Solution:

4. The *Hamming weight* of a string $x \in \{0,1\}^n$ is the number of positions equal to 1, where n is a power of 2. Devise a divide and conquer algorithm to build a circuit C_n with n input wires, $\lceil \log_2(n+1) \rceil$ output wires, and $O(n)$ total gates such that the Hamming weight of the input string is written in binary in the output wires. Prove your algorithm is correct and analyze the running time.

   Note: To solve this problem, you may use the fact that for any k, there is a circuit for adding two k-bit numbers with $O(k)$ gates. You may also assume that n is a power of 2.



   Hint: As stated above, use divide and conquer. The Master Theorem cannot be applied directly. First, find a way to use the Master Theorem to show that $T(n) \leq 2T(n/2) + O(n^{1/2})$ has an $O(n)$ solution. Then, justify why your proposed recurrence is better.

   Solution:

5. Let $A[1, \ldots, n]$ be an array that stores the heights of n people. All heights are distinct. Devise a divide and conquer algorithm to count the number of pairs of heights that are out of order, i.e. pairs (i, j) such that $i < j$ and $A[i] > A[j]$ that runs in $O(n \log n)$ time. Prove your algorithm is correct and analyze the running time. Here is an example of counting the number of out-of-order pairs:

   - $n = 5$. $A = (1.79, 1.72, 1.61, 1.55, 1.88)$. Then $(1, 2)$ is an out-of-order pair since $1 < 2$ and $A[1] = 1.79 > A[2] = 1.72$. There are 6 out-of-order pairs in total: $(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)$.

   Hint: Can you keep a count of the out-of-order pairs *while you are doing a merge sort*?

   Solution: