This homework is due on **October 2, 11am ET**.

Submit solutions to coding questions (Problem 2 and Problem 3) in Stepik, submit solutions to all other questions in Gradescope.

Please sign up at Stepik using the following link:
https://stepik.org/invitation/87519dc4727a0dbfd908e132a03323e5cd8b9fb2/

You can resubmit your code until it passes all tests, there is no limit on the number of attempts. You submit your solutions in any of the following programming languages `ASM32, ASM64, C, C#, C++, Closure, Dart, Go, Haskell, Java, Javascript, Julia, Kotlin, NASM, Octav, PascalABC.NET, Perl 5, PHP, Python 3, R, Ruby, Rust, Scala, Shell, Swift`, however starter files will be provided only for `Python, Java` and `C++`.

You are welcome to work with others, however you must explicitly list all collaborators and materials that you used. You must write up your own solution and your own code to every problem. See Georgetown University Honor System. When in doubt, ask the instructor what is allowed.

Problem 1 (King and Prisoner)**.** A clever prisoner was wrongly accused by the King and was given a death penalty. As a rule, the prisoner was given a last chance in which prisoner places n white and n black balls in two boxes with the only constraint that no box is empty. After this, the king picks one of the boxes (with probability 1/2 each), and picks a random ball from that box (each ball is picked with equal probability). If the ball happens to be black, then the prisoner is freed. Distribute the balls between the two boxes to maximize prisoner's chances. Prove that your solution is optimal.

Problem 2 (Two Missing)**.** Given an array that contains $n-1$ *distinct* numbers from $\{0, 1 \ldots, n-1, n\}$, find the two missing numbers.

Input: The first line contains an integer $n \geq 1$, the next $n-1$ lines contain $n-1$ distinct integers from the range $\{0, 1 \ldots, n-1, n\}$.

Output: If i and j are the two missing integers, then output $\min(i, j)$ and $\max(i, j)$ separated by a space.

Example 1:

Input:

```
5
4
0
1
3
```

The input contains all integers from $\{0, 1, 2, 3, 4, 5\}$ except for 2 and 5.

Output:

```
2 5
```

Problem 3 (Majority Element)**.** Given an array of length n that is *guaranteed* to have a majority element (an element appearing $> n/2$ times in the array), find the majority element.

Input: The first line contains an integer $n \geq 1$, and the next n lines contain n integers $a_1, \ldots, a_n$.

Output: If m appears more than $n/2$ in the input array $a_1, \ldots, a_n$, then output m. It is guaranteed that such m exists.

Example 1:

Input:

```
7
1
2
3
2
2
2
1
```

In this example $n = 7$. Since 2 appears $4 > 3.5 = n/2$ times in the input, 2 is the majority element.

Output:

```
2
```

Problem 4 (Boxes game)**.** In this exercise, we will design an algorithm for the following problem. Let N and S be integers, and N be a multiple of S. There are N closed boxes, each containing a bit (that is, a number from $\{0, 1\}$).

- In the preprocessing phase, the algorithm is allowed to check all boxes, and write down S bits in a notepad.
- In the query phase, for any integer $i \in \{1, \ldots, N\}$, you are allowed to read all bits written in the notepad, and open at most $(N/S - 1)$ boxes *not including* the box number i. After this, you should always correctly compute the bit in the box number i.

a. Design an algorithm for the case where $N = 2$ and $S = 1$. Here, given two bits x_1 and x_2, we want to preprocess them and compute one bit y, such that

- Given x_1 and y, we can compute x_2;
- Given x_2 and y, we can compute x_1.

b. Design an algorithm for the case where $S = 1$ but N is arbitrarily large. Here, given N bits $x_1, \ldots, x_N$, we want to preprocess them and compute one bit y, such that

- For every $i \in \{1, \ldots, N\}$, given y and the input bits $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N$, we can compute x_i.

c. Design an algorithm that works for all values of N and S (assuming that S divides N). Prove that your algorithm works correctly.

Problem 5 (**Extra credit.** Number of Maximal Independent Sets)**.** Recall that an independent set in a graph is a set of vertices, no two of which are adjacent. A maximal independent set is an independent set that is not a subset of any other independent set.

An n-vertex graph may have up to 2^n independent sets (think of an empty graph). In this exercise, we'll prove a stronger upper bound on the number of *maximal* independent sets in an n-vertex graph. For a vertex v of a graph G, $N_G[v]$ denotes the closed neighborhood of v, i.e., the set of all vertices of G adjacent to v and the vertex v itself.

a. Let v be a vertex of a graph G. Show that every maximal independent set must contain a vertex from $N_G[v]$. Show that if an independent set contains a vertex $u \in N_G[v]$, then it does not contain other vertices from $N_G[u]$.

b. Let $T(n)$ denote the maximum number of maximal independent sets in an n-vertex graph. Let d be the minimum degree of a vertex in G. Let v be a vertex of minimum degree in G. Use (a.) to obtain a recurrent upper bound on $T(n)$.

c. Solve the above recurrence (you can assume that n is a multiple of 3).

d. Construct a (not necessarily connected) graph where the number of maximal independent sets matches the bound you proved in (c.), thus showing that this bound is tight.

Problem 6 (**Extra credit.** SETH-hardness of Hitting Set)**.** In the Hitting Set problem, given a family $\mathcal{F}$ of subsets of $\{1, \ldots, n\}$, and a positive integer k, the goal is to decide where there is a set $H \subseteq \{1, \ldots, n\}$ of size $|H| \leq k$ such that H contains at least one element from each set in $\mathcal{F}$. Assume that $|\mathcal{F}| = \text{poly}(n)$.

a. Design an algorithm that solves the Hitting set problem in time $2^n \cdot \text{poly}(n)$.

b. Prove that under SETH, the Hitting Set problem cannot be solved in time 1.4^n.