This homework is due on **October 16, 11am ET**.

Submit solutions to coding questions (Problem 4) 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 (Linear Programming)**.** Consider the following linear programming problem:

$$\begin{aligned} &\text{maximize} && x + y \\ &\text{subject to} && 2x + y \leq 6 \\ & && 2y - x \leq 5 \\ & && x \leq 4 \\ & && y \leq 3 \\ & && x \geq 0 \\ & && y \geq 0 \end{aligned}$$

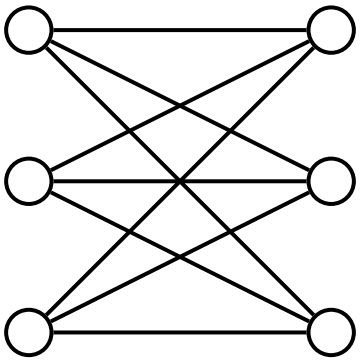
a. Plot the region of feasible solutions (the set of points satisfying all the above inequalities).

b. Find the optimal *real* solution of this linear programming problem. That is, find real x and y from the region of feasible solutions that maximize $x + y$. Explain why this solution is optimal.

c. Find an optimal *integer* solution of this problem. That is, find integer x and y from the region of feasible solutions that maximize $x + y$. Explain why this solution is optimal.

d. Formulate (but do not solve) an Integer Linear Programming instance for the following problem. There are four factories producing cars. They can produce up to $3, 2, 5$, and 4 cars per day, respectively. The amounts of materials these factories use to produce one care are $1, 1, 2$, and 2, respectively. Per one car, the four factories produce $2, 2, 3$, and 1 units of pollution. Producing a car takes $700, 500, 600$, and 400 hours of labor. We have 10 units of materials per day, 6000 hours of labor, we are allowed to produce 14 units of pollution per day, and we want to maximize the total number of cars produced by the four factories.

Problem 2 (Graph Colorings)**. a.** What is the chromatic number of the complete bipartite graph with three vertices on each side? Prove that your answer is optimal.



b. In Lecture 8, we proved that if the maximum vertex degree in a graph is Δ, then the graph can be colored using at most $\Delta + 1$ colors. Does there exist a graph where all vertices have degree at most 5, but the graph cannot be colored in 5 colors? If such a graph exists, prove that it cannot be colored with 5 colors. If every graph with maximum degree 5 can be colored with 5 colors, then prove this statement.

c. Let $\chi(G)$ be the chromatic number of a graph G (i.e., the *smallest* number of colors needed to color G). Prove that the number of edges in G is at least

$$\frac{\chi(G) \cdot (\chi(G) - 1)}{2}.$$

Problem 3 (Subsets with the Same Sum)**. a.** Construct a set of 4 integer numbers $A = \{a_1, a_2, a_3, a_4\}$ from $\{1, \ldots, 10\}$ such that all subsets of A have distinct sums.

b. Prove that every set of 6 integer numbers from $\{1, \ldots, 10\}$ contains two (non-empty) *disjoint* subsets with the same sum.

c. Construct a set of 30 arbitrarily large integer numbers such that all subsets of this set have distinct sums. Prove that all subsets of the set your constructed have distinct sums.

d. Prove that every set of 30 integer numbers from $\{1, \ldots, 1\,000\,000\}$ contains two (non-empty) *disjoint* subsets with the same sum.

Problem 4 (ILP- and SAT-Solvers. **Extra credit**)**. Part I. Subsets with the Same Sum**

In the previous problem we *proved* that every set of 30 integer numbers from $\{1, \ldots, 1\,000\,000\}$ contains two disjoint subsets with the same sum, but *finding* such subsets is a hard problem. In this part of the problem, we will use an ILP solver to efficiently find subsets with the same sum. See Lecture 11 and https://docs.python-mip.com/en/latest/examples.html for examples.

Input: The input consists of 30 integers from $[1, 1\,000\,000]$ separated by spaces.

Output: The output should consist of two lines, each line contains the numbers in one of the subsets. The two subsets must be non-empty, disjoint, and have the same sum.

Example 1:

Input:

```
2050 452997 209798 194439 117263 455185 66707 296983 36892
410783 395683 36650 134186 270132 497980 341183 461250
373703 190796 376148 303830 345817 245851 178271 498801
477811 453749 481656 40313 69755
```

Output:

```
194439 66707 270132 453749 40313
452997 36892 36650 498801
```

Example 2:

Input:

```
123009 226056 405641 82316 82198 294681 428953 363675 209945
390557 388257 450987 263227 89916 484543 97216 275138 252101
88266 181326 390610 50900 58080 401380 434792 255720 309613
77549 329710 334205
```

Output:

```
88266 181326 434792 255720 77549 334205
123009 82198 89916 97216 275138 252101 50900 401380
```

Part II. Sudoku.

Use the SAT solver pycosat (https://pypi.org/project/pycosat/) to solve given Sudoku (https://en.wikipedia.org/wiki/Sudoku) puzzles.

Input: The input contains nine lines, each line has nine characters. The characters are either numbers from 1 to 9, or "*" symbols which denote empty cells.

Output: If the puzzle doesn't have a solution, output -1. Otherwise output any solution to the given puzzle.

Example 1:

Input:

```
8********
**36*****
*7**9*2**
*5***7***
****457**
***1***3*
**1****68
**85***1*
*9****4**
```

Output:

```
812753649
943682175
675491283
154237896
369845721
287169534
521974368
438526917
796318452
```

Example 2:

Input:

```
82*******
**36*****
*7**9*2**
*5***7***
****457**
***1***3*
**1****68
**85***1*
*9****4**
```

Output:

```
−1
```

Problem 5 (Exponential-time Algorithms. **Extra credit**)**. Part I. Graph Coloring.**

a. Beigel and Eppstein (J. Algorithms 2005) designed an algorithm that in time 1.33^n checks if an n-vertex graph admits a 3-coloring. Use this algorithm (in a black-box way) and the Problem 5 from Homework 2, to design an algorithm that checks if an n-vertex graph admits a 4-coloring in time $(2-\varepsilon)^n$ for a constant $\varepsilon > 0$.

b. In Lecture 8, we saw how to solve the chromatic number problem in time 3^n. Use the Problem 5 from Homework 2 to improve the running time of that algorithm to $O^*(1+3^{1/3})^n$.

Part II. 3-SAT.

a. Let ϕ be a satisfiable 3-SAT formula with n variables, and $x \in \{0,1\}^n$ be an assignment to the variables of ϕ (that doesn't satisfy ϕ). Assume that the Hamming distance between x and a satisfying assignment of ϕ is at most d. Design an algorithm that runs in time $O^*(3^d)$ and finds a satisfying assignment of ϕ.

b. Give an algorithm that solves 3-SAT in time $O^*(3^{n/2})$.

c. Design a randomized algorithm that solves 3-SAT in time $O^*(1.5^n)$ with high probability. For this, pick k random assignments $x_i \in \{0,1\}^n$, and check if there are satisfiable assignments in Hamming balls of radius $n/4$ centered at x_i.