1. (Exponential-Time Coloring) In the Github repository for PS8, we have given you basic data structures for graphs (in adjacency list representation) and colorings, an implementation of the Exhaustive-Search $k$-Coloring algorithm, and a variety of test cases (graphs) for coloring algorithms.

(a) Implement the $O(n + m)$-time algorithm for 2-coloring that we covered in class in the function `bfs_2_coloring`, verifying its correctness by running `python3 -m ps8_tests 2`.

(b) Implement the $O(1.89^n)$-time algorithm for 3-coloring that you studied in Active Learning Exercise 5 in the function `iset_bfs_3_coloring`, also verifying its correctness by running `python3 -m ps8_tests 3`.

(c) Finally, implement the reduction from 3-coloring to SAT given in class in the function `sat_3_coloring`, producing an input that can be fed into the SAT Solver Glucose, and verify its correctness by running `python3 -m ps8_tests 3`.

(d) Compare the efficiency of Exhaustive-Search 3-coloring and your implementations from Part 1b and Part 1c using `ps8_experiments`. In the experiments file, we’ve provided code to generate two types of graphs: lines of rings and clusters of independent sets. Across all of the tests, identify the largest instance each algorithm is able to solve (within a time limit above 10 seconds of your choice) and the smallest instance each algorithm is unable to solve (again within the same time limit). In addition to the time limit you chose, your answer should include 6 numbers: two values of $n$ for each of the three algorithms. Briefly discuss and try to explain your findings.

2. (Reductions to SAT) Consider the following problem. From Harvard’s $n$ CS concentrators (e.g. $n = 400$), we want to form a team of exactly $k$ students (e.g. $k = 30$) to represent Harvard in a new programming competition. The programming competition problems may require expertise in any of $m$ different programming languages (e.g. $m = 100$). But each of the CS concentrators only knows a few different programming languages, with a different set per person. So we want to try to find $k$ Harvard CS concentrators such that between them, they know all $m$ languages. Show how this problem can be efficiently reduced to solving a SAT instance on $kn$ variables and $m + O(kn^2)$ clauses. Prove the correctness of your reduction and analyze its runtime.