Problem 1-1. Counting and Sets [10 points]

Suppose the letters a, b, c, d have proper positions 1,2,3,4 i.e. the correct sequence is $abcd$. Write down all the deranged sequences (where none of the letters are in their proper position). Find a generalised combinatorial expression for n letters and use it to verify your answer.

Problem 1-2. Probability Practice: Dice Rolling [35 points]

Asami and Bolin are playing a dice game in which a pair of dice is rolled repeatedly. Asami wins if a 6 is rolled before any number greater than or equal to 9, and Bolin wins if any number greater than or equal to 9 is rolled first. We will find the probability that Asami wins the game.

(a) *[Warm-up] What is the probability that a 6 is rolled on any given roll of the pair of dice? What is the probability that any number greater than or equal to 9 is rolled on any given roll of the pair of dice?*

(b) *[Warm-up] Let E_n denote the event that a 6 occurs on the n^{th} roll and neither 6 nor any number greater than or equal to 9 occurs on any of the first $(n-1)$ rolls. Compute $P(E_n)$.*

(c) [10 points] Compute $\sum_{n=1}^{\infty} P(E_n)$ and argue rigorously that it is the desired probability. **HINT:** $\sum_{n=0}^{\infty} r^n = \frac{1}{1-r}$ for $|r| < 1$.

(d) [25 points] Now, suppose that in a run of this game, the dice are rolled ten times with neither a 6 nor any number greater than or equal to 9 appearing. Given this, let X be the sum of these ten rolls. We would now like to find an upper bound for the following probability $P[|X - \frac{1220}{21}| \geq 10]$.

1. [3 points] Given that each of these ten rolls results in neither a 6 nor any number greater than or equal to 9, enumerate the possible totals for each roll and their respective probabilities.
2. [5 points] For these ten rolls, let X_i be the number on the face of the die for roll i. Compute the expected value of X_i.
3. [5 points] Compute $\mathbb{E}[X]$ where X is the sum of these ten dice rolls.
4. [8 points] Compute the variance of X, where X is again the sum of these ten dice rolls.
5. [4 points] Use Chebyshev's inequality to find an upper bound for $P[|X - \frac{1220}{21}| \geq 10]$.

Problem 1-3. Su's Symmetry Studies [50 points]

Darth Su, physics and computer science experimentalist extraordinaire, is on track to perform an experiment that will expose a flaw in one of the most fundamental physics and computer science theories of the time—that of parity bit symmetry in the CPTLRS symmetry theory. To demonstrate the violation of the parity bit symmetry, she deposits a thin surface layer of cobalt-60 atoms onto a crystal of cerium-magnesium nitrate, and they decay to produce bits of 0 or 1, which she hopes to show are imbalanced in count and thus asymmetric.

The deposition of atoms is thin enough that the set of atom positions G can be modeled as points on the plane, where the location of each atom $g_i \in G$ is represented by the tuple (x_i, y_i) with each coordinate measured in angstroms. However, Su finds that in order to accurately measure the deviation from perfect symmetry, each atom must be at least 1 angstrom away from every other atom. Su, eager to change the course of physics and computer science, reaches out to you for help.

Your job as a 1203 student is to help Su determine whether the set of atoms G violates this property. More formally, propose an algorithm to check whether there exists a pair of points $g_i, g_j \in G$ such that $||g_i - g_j|| < 1$ (meaning that their Euclidean distance is less than 1), prove the correctness of this algorithm, and analyze its runtime.

(a) *[Warm-up]*

Consider first a simplification of the problem in which all of the atoms happen to lie on a single line, and each atom can be characterized by only an x-coordinate x_i.

Propose any $O(n \log n)$ solution to determine whether there exists a pair of atoms less than 1 angstrom apart. If such a pair exists, your algorithm should return any example pair, and if no pair exists, your algorithm should return false. Your explanation need only be 2-3 sentences long, and you can provide your algorithm without proof of correctness or runtime analysis.

(b) *[Warm-up]*

Now, we return to the two-dimensional case described in the problem statement.

Give an $O(n^2)$ algorithm to determine whether there exists a pair of atoms less than 1 angstrom apart. If such a pair exists, your algorithm should return any example pair, and if no pair exists, your algorithm should return false. Your explanation need only be 2-3 sentences long, and you can provide your algorithm without proof of correctness or runtime analysis.

(c) [5 points]

Suppose the planar deposition is divided into a grid of squares, where each square has dimensions $\frac{1}{2} \times \frac{1}{2}$. Why must you reject G if two atoms are in, or on the boundary of, the same square?

(d) [20 points]

Propose an algorithm to determine whether there exists a pair of atoms less than 1 angstrom apart, providing a proof of correctness and a runtime analysis. For full credit, your algorithm should run in $O(n \log n)$ time with respect to $n = |G|$, the number of atoms.

Hint: Use divide-and-conquer with part (c).

(e) [25 points]

Upon further analysis, Su realizes that the deposition of atoms is not quite as thin as she expected, so she can no longer make the assumption that the atoms lie in a single plane. However, Su must still ensure that none of the atoms are too close to other atoms, lest the experiment fail.

Let A be the set of all atom positions in this three dimensional setup, such that every atom position $a_i \in A$ has 3D coordinates (x_i, y_i, z_i), each measured in angstroms.

Give a solution to determine whether there exists a pair of atoms $a_i, a_j \in A$, $i \neq j$ such that the distance between the two atoms *in the 3D space* is less than 1 angstrom. If there is such a pair, your algorithm should return an example pair, and if there is no pair, your algorithm should return *false*. Prove the correctness of your algorithm, and analyze its runtime. For full credit, your algorithm should run in $O(n \log^2 n)$ time with respect to $n = |A|$, the number of atoms.

Hint: Use your solution in part (d) as part of the divide-and-conquer process.

Problem 1-4. Land It or Not [20 points]

We discussed the following situation in class. If you paid attention, you should find it ridiculously easy to get full points on this question.

Imagine you're in charge of managing ground traffic at a busy airport. Pilots request landing times, and you must decide whether to approve or decline based on the following constraint: if the requested landing time is n_1 and $\exists n_2$ which is pre-approved, then $|n_1 - n_2|$ should be greater than buffer time, k. For simplicity, let the time shared be in the format: $n \in \mathbb{Z}_+$. Pilots won't necessarily request their time in the order in which they land, so you may get requests in any order (like 5, 10, 3 for example). If the request is approved, it should then be accepted and stored in a manner such that you can print out a sorted order of landings.

Example: If a pilot requests to land at time 10 and there's a pre-approved landing time of 7, and the current buffer time is $k = 5$, then the request should be declined.

(a) [10 points] You need to design a system that efficiently handles landing requests and keeps a record of approved requests. Can you use arrays to keep track of this? If so, briefly explain how you would do so.

(b) [10 points] Explain how you could also use binary search trees for this. Compare the efficiency of the BST solution and the array solution.

Problem 1-5. Enchanted Forest [5 points]

Once upon a time, in a mystical land, there was an ancient forest filled with enchanted creatures and magical flora. At the heart of this forest stood a magnificent tree, known as the "Tree of Wonders." Legend had it that this tree held unimaginable secrets and was guarded by the mystical creatures dwelling within.

However, a powerful curse had befallen the forest, causing it to wither and lose its magic. The key to breaking this curse lay in a special node of the Tree of Wonders, known as the "Curse Node". The curse node stores the C-th smallest value of the Tree of Wonders. It was said that removing this node would restore the forest's vitality. However, it is possible that while removing the curse node a new value is added to the Tree of Wonders, thus changing the value of the C-th smallest element!

Your task is to restore the Enchanted Forest. Once you have successfully completed your quest, you shall present the final cursed node you extracted from the tree. Explain how you would keep track of the C-th smallest node at any given time as numbers are being added to the Tree of Wonders. Make sure you explain any data structures you would use, and how they help you to solve the problem efficiently.

May the magic of the forest guide your hand!

Problem 1-6. Coding Section [20 points]

We will now implement the data structures and algorithms we have discussed in Problems 1-4 and 1-5. To set up this assignment in GitHub classroom click here. Once your repository is set up, clone it to your local machine.

To compile code in this assignment properly run `make` from the assignment's directory. This will output an executable `asmnt1` that requires an integer input parameter. For example, to execute this assignment you could run `./asmnt1 5`. To clear the compiled files you can run `make clean`, it might make sense to do this before pushing your answers. The `make clean` command might fail on windows – just ignore it.

The integer `5` acts as a seed to generate random numbers using the Mersenne Twister algorithm. Any given seed will always generate the same set of random numbers in the same order, which will allow you to test the code you write easily, and allow us to test your code with our own set of random numbers. Use the same 15 random numbers in the order in which they are stored in the array to add to your data structures.

(a) [7 points] Implement the BST solution to the runway problem in `C`. We have provided you with a few function primitives to get started, but feel free to code up any additional helper functions you need along the way. You should generate 15 random numbers using the functions provided and then use the BST solution to accept or reject them with $k = 12$. At each stage you should print the sorted order of the tree on a new line in the format `Inorder: w x y z` and so on where the letters are the integers for each landing time that has been accepted.

(b) [3 points] The second plane that sent you a request didn't take off. Remove its landing time from the tree and print the tree again. You should output `Updated: w x y z` and so on as earlier on a new line.

(c) [10 points] Implement your solution for Problem 1-5. We have set $C = 4$ for this implementation. After inserting each element, print the new C-th smallest element on a new line in the format `The 4th smallest element is x` where `x` is the element.