This assignment has 6 questions, for a total of 50 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big oh and running times</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Eager or not?</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Square vs. Multiply</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Graph basics</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Background: Probability</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Array Sums</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: Big oh and running times

(a) Write down the following functions in big-oh notation:
1. \( f(n) = n^2 + 5n + 20 \)
2. \( g(n) = \frac{1}{n^2} + \frac{2}{n} \)

(b) Consider the following algorithm to compute the GCD of two positive integers \( a, b \). Suppose \( a, b \) are numbers that are both at most \( n \). Give a bound on the running time of \( \text{Gcd}(a, b) \). (You need to give a formal proof for your claim.)

```
Algorithm 1 \text{Gcd}(a, b)
if (a < b) return \text{Gcd}(b, a);
if (b = 0) return a;
return \text{Gcd}(b, a\%b); (Recall: \( a\%b \) is the remainder when \( a \) is divided by \( b \))
```

Question 2: Eager or not?

Imagine a world in which Moore’s law holds perfectly (every two years, a computer twice as fast arrives). A graduate student enters grad school and realizes she needs to finish a certain simulation in order to graduate. Using the current machines, it would take her 7 years for the computation to complete. Naturally, she wishes to graduate as quickly as possible. Should she start computation right away? If not, what is her best strategy?

Question 3: Square vs. Multiply

Suppose I tell you that there is an algorithm that can square any \( n \) digit number in time \( O(n \log n) \), for all \( n \geq 1 \). Then, prove that there is an algorithm that can find the product of any two \( n \) digit numbers in time \( O(n \log n) \). [Hint: think of using the squaring algorithm as a subroutine to find the product.]

Question 4: Graph basics

Let \( G \) be a simple undirected graph. Prove that there are at least two vertices that have the same degree.

Question 5: Background: Probability

(a) Suppose we toss a fair coin \( k \) times. What is the probability that we see heads precisely once?

(b) Suppose we have \( k \) different boxes, and suppose that every box is colored uniformly at random with one of \( k \) colors (independently of the other boxes). What is the probability that all the boxes get distinct colors?

(c) Suppose we repeatedly throw a fair die (with 6 faces). What is the expected number of throws needed to see a ‘1’? How many throws are needed to ensure that a ‘1’ is seen with probability > 99/100?

Question 6: Array Sums

Given an array \( A[1 \ldots n] \) of integers, find if there exist indices \( i, j, k \) such that \( A[i] + A[j] + A[k] = 0 \). Can you find an algorithm with running time \( o(n^3) \)? [NOTE: this is the little-oh notation, i.e., the algorithm should run in time \( < cn^3 \), for any constant \( c \), as \( n \to \infty \).] [Hint: aim for an algorithm with running time \( O(n^2 \log n) \).

---

1I.e., there are no self loops or multiple edges between any pair of vertices.