This homework is due on **09/30/24, 12:30pm ET**.

Academic Integrity. You can work in groups of two or three people, 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.

The use of AI text generating tools (such as ChatGPT) at any point when working on the assignments will be treated as academic dishonesty.

You may only discuss problems with your group members and the course staff. The only materials you can reference when working on these problems are the textbooks, course notes, and lectures from the current semester of this course. You may not reference any other online materials or solutions from other semesters.

Coding Questions. Submit solutions to coding questions (Extra Credit Problem 5) in Cogniterra, submit solutions to all other questions in Canvas. You can typeset your solutions in LaTeX (for example, using this template: https://www.overleaf.com/read/gjdgtxxpwkkj).

Please sign up on Cogniterra using the following link:
https://cogniterra.org/invitation/c15f17682a6170840ecde408154e8655b771c981/

You can resubmit your code until it passes all tests, there is no limit on the number of attempts. You can submit your solutions in any of the following programming languages `C, C#, C++, Go, Haskell, Java, Javascript, Kotlin, Python 3, Ruby, Rust, Scala, TypeScript`, however starter files will be provided only for `Python, Java`, and `C++`.

Problem 1 (Asymptotic Notation, **10pt**)**.**

a (5pt) Which of the following statements are correct? Provide brief explanations.

1. $3n + 5 = O(n)$
2. $\sqrt{n} + n + n^2 = O(n)$
3. $n^3 + n^2 = O(n^2)$
4. $n^3 + n^2 = O(n^4)$
5. $(\log n)^{10} = O(\sqrt{n})$
6. $2^{n+10} = O(2^n)$
7. $2^{2n} = O(2^n)$

b (5pt) Rank the following functions by order of growth. Provide brief explanations.

1. $5\sqrt{n}$
2. $5 \log_2 n + 3$
3. $n - 3$
4. $0.5n^3 - n^2$
5. $2^{n/2}$
6. $n^{3/7}$
7. $n \log_2 n$
8. 1
9. $\sqrt{\log n}$

Problem 2 (Master Theorem, **10 pt**)**.** Use the Master Theorem to solve the following recurrences, provide detailed solutions for each item.

- $T(n) = 2T(n/2) + O(n)$
- $T(n) = 4T(n/2) + O(n)$
- $T(n) = 2T(n/4) + O(n)$
- $T(n) = 2T(n/4) + O(\sqrt{n})$
- $T(n) = 9T(n/3) + O(n^2)$
- $T(n) = 7T(n/3) + O(n)$
- $T(n) = 7T(n/3) + O(n^2)$

Problem 3 (Smallest Missing Number, **10pt**)**.** In this exercise, the task is to design an $O(\log n)$-time divide-and-conquer algorithm for the following problem. Provide pseudocode, prove its correctness, and analyze its running.

Given a sorted array of n distinct non-negative integers, find the smallest non-negative integer that does not appear in this array. In particular, if the input array of length n contains all integers from $\{0, \ldots, n-1\}$, output n.

Input: A sorted (in ascending order) array $A[1], \ldots, A[n]$ of distinct non-negative integers.

Output: The smallest non-negative integer that does not appear in this array.

Example 1:

Input:

```
0 1 2 3 5 8 9
```

The input contains the first four non-negative integers: 0, 1, 2, and 3, and does *not* contain 4.

Output:

```
4
```

Example 2:

Input:

```
1 2 3 7
```

The input does not contain the smallest non-negative integer 0.

Output:

```
0
```

Example 3:

Input:

```
0 1 2 3
```

The input of length $n = 4$ contains all integers from 0 to $n - 1 = 3$, so the smallest missing non-negative integer is 4.

Output:

```
4
```

Problem 4 (2-Sum, **10+2 pt**)**.** In each part of this exercise, the goal is to design an efficient algorithm, provide pseudocode, prove its correctness, and analyze its running. You can use procedures that we designed in class, you do not need to repeat their pseudocode, but you should explain what they do and recall their running time.

a (10pt) Given an array A of n integers, and another integer x, check whether or not there exist two elements in A whose sum is x (you may not use the same element of A twice). If there exist distinct indices i and j such that $A[i] + A[j] = x$, output i and j for any such pair, otherwise output -1.

For partial credit, give an algorithm solving the problem in time $O(n^2)$, for full credit provide a more efficient algorithm.

Example 1:

Input:

```
A=[8  7  8  9  15]
x = 16
```

There are two solutions, $A[1] + A[3] = 8 + 8 = 16$ and $A[2] + A[4] = 7 + 9 = 16$, outputting any of this pairs would be correct.

Output:

```
2  4
```

Example 2:

Input:

```
A=[7  4  4  12  1]
x = 4
```

No pair of elements from A sums up to 4, therefore the output must be -1.

Output:

```
−1
```

b (Extra credit, 2pt) Now the goal is to check if there are **three** distinct indices i, j, k such that $A[i] + A[j] + A[k] = x$.

For partial credit, give an algorithm solving the problem in time $O(n^2 \log n)$, for full credit provide a more efficient algorithm.

Problem 5 (**Extra Credit 0+4pt).** In each part of this exercise, the goal is to implement an efficient divide-and-conquer algorithm, and submit it on Cogniterra.

a (Extra credit, 1pt) Binary Search.

b (Extra credit, 1pt) Maximum Subarray Problem.

c (Extra credit, 1pt) Duplicates.

d (Extra credit, 1pt) Anagrams.