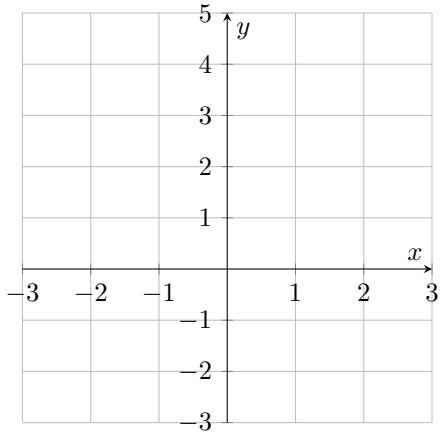
CSCI 246: Assignment 5
Due: April 8, 2026

Name: _____

Problem 1 (10 points). Let $f : \mathbb{R} \to \mathbb{R} = \{(x, x^3 - 2x + 1) : x \in \mathbb{R}\}$.

A. Is f a function? Why or why not?



B. Graph f on the domain $[-2, 2]$ (use plot to right).

C. What is the domain, image, and co-domain of f?

D. Is f one-to-one (injective)? why or why not?

E. Is f onto (surjective)? Why or why not?

F. Does f have an inferse function f^{-1}? Why or why not?

Problem 2 (10 points). For each pair of expressions, determine which grows faster as $n \to \infty$. Justify your answer.

A. n^2 and $n \log n$

B. 2^n and $n!$

C. $\log n$ and $\sqrt{n}$.

D. $n^{100^{100}}$ and 3^n

E. $n^3 + n \log n^{100}$ and $n^3 \log n$

Problem 3 (10 points). For each pair of functions $f(n)$ and $g(n)$ below. Determine if:

1. $f(n) \in \Omega(g(n))$,
2. $f(n) \in O(g(n))$, and
3. $f(n) \in \Theta(g(n))$.

A. $f(n) = n^2 + n \log n$ and $g(n) = n^2$

B. $f(n) = \log n^{1000}$ and $g(n) = \log n$

C. $f(n) = (n!)(n!)$ and $g(n) = n^{2n}$

D. $f(n) = \log n$ and $g(n) = n^{1.00001}$

E. $f(n) = 2^n$ and $g(n) = 3^n$

Problem 4 (10 points). Suppose you have two algorithms for sorting objects. The first algorithm has a runtime of $f(n) = n^2 + n + 1$. The second sorting algorithm takes $g(n) = 10n \log n + 5n + 10$. Explain when you might want to use the first algorithm over the second, and when you might want to use the second algorithm over the first. (Hint: compute $f(n)$ and $g(n)$ for $n = 10, 20, 30, 40, ...$).

Problem 5 (10 points). Provide functions $f(n)$ and $g(n)$ such that $f(n) \notin O(g(n))$ and $g(n) \notin O(g(n))$; i.e., the order of growth of $f(n)$ and $g(n)$ are incomparable.

Problem 6 (10 points). Consider the following code snippet.

```
# accepts arrays containing n elements
def sort(arr : Array[n]) -> Array[n]:
  i = 0
  while i < n - 1:
    min_ind = i

    j = i + 1
    while j < n:
      if (arr[j] < arr[min_ind]):
        min_ind = j
      ++j
    ++i

    swap(arr[i], arr[min_ind])
```

You may assume that comparisons and swaps are the only elementary operators of concern.

A. Come up with $B(n)$ a function that captures the best-case runtime complexity of the sorting algorithm in terms of n the size of the input array. When is the best case complexity realized? What is the order of growth of $B(n)$?

B. Come up with $W(n)$ a function that captures the worst-case runtime complexity of the sorting algorithm in terms of n the size of the input array. When is the worst case complexity realized? What is the order of growth of $W(n)$?

C. What is the average-case runtime complexity of the algorithm? Justify your answer.