Math Problem 1

(1) $f(x) = -x^2$

Taking derivative of $f(x)$, we get

$f'(x) = -2x$

For $x < 0$, $f'(x)$ is negative

$x > 0$, $f'(x)$ is ~~negative~~ positive

Hence, $f(x)$ is decreasing for x. It is not eventually non decreasing because for all $x > 0$, it keeps decreasing and never is stable.

(2) $f(x) = x^2 + 2x + 1$

Taking derivate, $f'(x) = 2x + 2$

For $x < -1 \Rightarrow f'(x) < 0$

$x \geq -1 \Rightarrow f'(x) \geq 0$

It ~~stops decreasing~~ is not increasing, but eventually non decreasing

(3) $f(x) = x^3 + x$

$f'(x) = 3x^2 + 1$

For $-\infty < x < \infty$, $f'(x)$ is positive

Hence, it is increasing everywhere, also eventually increasing

Math Problem 2

(1) $f(x) = 2x^2$, $g(x) = x^2+1$

$\Rightarrow O(x^2)$ $\qquad \Rightarrow O(x^2)$

They are asymptotically equal.

(2) $f(x) = x^2$, $g(x) = x3$

Here, x^3 grows fastere than x^2,

(3) $f(x) = 4x+1$, $g(x) = x^2-1$

x^2 grows faster than $4x$

Hence, $f(x)$ grows no faster than $g(x)$, not vice versa.

Problem 1

GCD Algorithm Code:

```java
public class GCDAlgorithm {
    1 usage
    public static int findGCD(int m, int n) {
        while (n != 0) {
            int temp = n;
            n = m % n;
            m = temp;
        }
        return m;
    }
    public static void main(String[] args) {
        System.out.println("GCD of 12 and 8: " + findGCD(12, 8));
    }
}
```

Problem 2:
Brute Force Solution:

```java
no usages
public class SubsetProblem {
    no usages
    public static List<Integer> subsetSum(int[] set, int k) {
        return subset(set, 0, k, new ArrayList<>());
    }

    3 usages
    private static List<Integer> subset(int[] set, int i, int k, List<Integer> subset) {
        if (k == 0) return new ArrayList<>(subset);
        if (i == set.length || k < 0) return null;
        subset.add(set[i]);
        List<Integer> include = subset(set, i + 1, k - set[i], subset);
        if (include != null) return include;
        subset.remove(subset.size() - 1);
        return subset(set, i + 1, k, subset);
    }
}
```

Problem 3: Greedy Strategy

Consider the sorted set of numbers: {1, 5, 7, 10, 15}, and the target sum is 12.

- **Greedy Output:** {1, 5} (Total = 6, which is incorrect!)
- **Correct Subset:** {5, 7} (since 5 + 7 = 12)

This shows that the greedy strategy doesn't always work. Here's why:

- The greedy approach fails because it picks the smallest numbers first, which blocks better combinations.
- In this case, by choosing 1 and 5, the algorithm misses the combination of 7 and 5, which would give the exact sum.
- The strategy of picking smaller elements doesn't always lead to the optimal solution, especially in cases where larger elements might be better suited for reaching the target sum.

Problem 4:

It really depends on the situation. If the subset T minus the smallest number (sn-1) still adds up to the target sum (k′), then removing sn-1 can still lead to a valid solution. However, sometimes sn-1 is crucial to reaching the target sum, and removing it could break the solution. So, removing sn-1 from the set T doesn't guarantee we still have a valid solution.