CIT-594-HW1:

PART I:

1. **Consider an array A of size n >= 2. The array contains numbers from 1 to n-1 (both inclusive) with exactly one number repeated. Describe an O(n) algorithm for finding the integer in A that is repeated.**

if integer is n0, we can run a full loop to search for that:

```
int[] B = new int[n-1];                    // All elements initialized to 0.
```

/* use a n-1 elements array B to find the repeated number, with its indices equal to each number-1 in A, the element with index number equal to (the repeated number -1) should be in loops twice. */

```
for (i = 0; i < A.length; i++){
   if(B[A[i]-1]>0) return A[i];           //return the repeated number
   else B[A[i]-1]++;
 }
 return -1;                                // return -1 when error occurs.
```

Since this algorithm runs a n elements loop once, it is O(n).

2. **What is the order of growth for the following (using the Big-Oh notation)? (5 points each)**

a. O(n), for a large n >> 5, the inner loop complexity is closer to O(1), which means each loop has a complexity of O(1), with total n loops. So it is O(n).

b. For No. i loop in the outermost loops has (i*i-1) operations, and the innermost loops will executed every i loops, with total i-1 times, and each time will cause a j times operations. So we have:

$$\sum_{i=1}^{n-1}\left\{(i^2-1)+i\times\sum_{j=1}^{i-1} j\right\}=O(n\times(n^2+n^3))=O(n^4)$$

3. **Consider the following two functions and the claim that f(x) is O(g(x)): f(x) = 3x^4 + 5x^3 + 17x^2 + 13x + 5; g(x) = x^4 Prove whether the claim is true or false using the definition of Big-Oh. (5 points)**

The complexity of f(x) is O(x^4), and g(x) has the same complexity. Say, we can assume n0=6, c = 0.1, s.t. for all x>n0, c× f(x)<g(x).

to prove that, we can calculate the value of c×f(x)/g(x) = 0.1× (3+5/x+17/x^2+13/x^3+5/x^4), when x>6, this value always less than 1 (when x=6, value < 1, and this value decreases with x increases), so that c× f(x)<g(x) is always true.