1a. **public** **static** **int** square (**int** n) {

**if**(n == 1) **return** 1;

**return** *square*(n-1) + (2\*(n-1)) + 1;

}

1b. **Theorem:** P(n): “if, then ”

**Proof:** prove by mathematical induction

**Base case:** plug in n = 1 to check that P(1) is true

,

,

,

**Inductive step:** Assume , P(n) => P(n+1)

Inductive hypothesis: P(n): “if , then ”

1. Assume that P(n) is true;  
    ,   
    —①
2. If P(n) is true, will P(n+1) be true;  
    , ,   
    —②
3. Replace ① into ② ;

**Conclusion:** From principle of mathematical induction, this implies that P(n) is true for any positive integer

2. **Theorem:** S(n): “ An error-detecting set of strings of length n using digits 0, 1, and 2, cannot have more than 3­n-1 strings for any n > 1 “

**Proof:** prove by contradiction.

1. Assume that the set of error-detecting strings has more than 3n-1 elements. And let say that the set has A elements
2. I observe that the error-detecting strings are formed by selections with replacement of 0, 1, and 2 with length equal to n, so number of the elements from the selections with replacement that are not the error-detecting strings (let me call them non-error-detecting strings) should be equal to the total number of the elements from the selection subtract with the number of elements in error-detecting set.
3. Let assume that , how many non-error-detecting strings will be in the selection with replacement

NED = {01, 02, 10, 12, 20, 21}, n(NED) = 6

1. Since there are 6 non-error-detecting strings, then there will be error-detecting strings
2. From step 4 I see that the first assumption (set of error-detecting strings has more than 3n-1 elements) is false
3. This mean that the set of error-detecting strings cannot have more than 3n-1

3. **Theorem:** S(n): “ for any

**Prove:** prove by mathematical induction

**Base case:** ,

**Inductive step:** Assume , P(k) => P(k+1)  
 Inductive hypothesis: P(k): “ “

1. Assume that P(k) is true for any ;  
    —①
2. If P(k) is true, will P(k+1) be true;  
    —②  
   replace ① into ②;

**Conclusion:** From the principle of mathematical induction, this implies that S(n): “ “ is true for any integer greater than or equal to 0

4.

5. **Theorem:** P(n): “ => is . “

**Proof:**

1. Let ; when a satisfy the inequality
2. Replace into ;
3. From step 2, I see that is multiply with some value .

**Conclusion:** is when and

6. **Theorem:** if is and is a function whose value is never negative, then is .

**Proof:**

1. is :
2. Assume that is a constant k.
3. Since the k is always >= 0, and is , then is .

7. **Theorem:** If is , then is .

**Proof:** prove by direct proof.

1. is , This mean that for some value C, .
2. There are 2 cases for : ① ②
3. Consider ①. Since the theorem state that is then, the first case is true.
4. Consider ②. The second case is obviously true from the definition of Big-O notation: ; when C > 1
5. From step 3 and step 4, the result of is always