CS240: Solving Recurrences

1. Consider the following recurrence:

$$\boxed{\begin{array}{l} T(1) = 5 \\ T(n) = 5 + T(n-1) \end{array}}$$

   a. Solve the recurrence using backward substitution**:** **T(n) =** _____

   b. Check your answer by applying both the recurrence and your solution:

Apply the **recurrence**:	Apply your closed-form **solution**:
T(1) =	T(1) =
T(2) =	T(2) =
T(3) =	T(3) =
T(4) =	T(4) =
T(5) =	T(5) =

2. Analyze the following code:

```
public static int fun(int n) {
    if (n == 0) {
        return 5;
    } else {
        int sum = 1;
        for (int i = 0; i < 4; i++) {
            sum *= 2;
        }
        return sum * fun(n - 1);
    }
}
```

a. First, develop a recurrence that counts the number of **multiplication operations**:

Initial condition(s):	**Recurrence relation:**

b. Next, use backward substitution to find a closed-form solution: **T(n) =** _____

c. Check your answer!

3. Analyze the following pseudocode:

```
function magic( nums : list of ints ):
  if nums.size == 1:
    return nums[0]

  sum = 0
  for each index i in nums:
    sum += nums[i]

  remove the last item from nums so it is one smaller

  return sum + magic(nums)    // WATCH OUT! Nums is now one smaller
```

a. First, develop a recurrence that counts the number of **indexed accesses (underlined)**:

Initial condition(s):	**Recurrence relation:**

b. Next, use backward substitution to find a closed-form solution: **T(n) =** _____

c. Check your answer!

4. Analyze the following code:

```
public static int fun3(int[] numbers) {
    return fun3(numbers, numbers.length);
}

private static int fun3(int[] numbers, int index) {
      if (index <= 1) {
            return 3;
      }

      for (int i = 0; i < 4; i++) {
            fun3(numbers, index / 2);
      }

      int sum = 0;
      for (int i = 0; i < index; i++) {
            for (int j = 0; j < index; j++) {
                  sum += numbers[i] + numbers[j]; // count this
            }
      }
      return sum;
}
```

Develop a **recurrence** and use the method of **backward substitution** to create a formula for the number of times the underlined code is executed in *fun3* for different lengths of *numbers*.

(For the purpose of analysis, you may assume that the length of *numbers* is a power of 2)

T(n) = _____