(1) Consider the following function, which returns all the strings of length $n$ over a given alphabet.

```python
def all_strings(alphabet, n):
    '''
    Given a non-empty list of characters alphabet and a positive natural number n, returns a list of all the strings of length n over the alphabet.
    '''
    if n == 1:
        result = alphabet
    else:
        t = all_strings(alphabet, n - 1)
        result = []
        for s in t:
            for a in alphabet:
                result.append(a + s)
    return result
```

(a) Let $k$ be the size of `alphabet`. Write a recurrence $T_k(n)$ for the size of `result`. So $T_k(n)$ is the number of strings of size $n$ over an alphabet of size $k$. Write 1-2 sentences explaining why the recurrence is correct.

(b) Use repeated substitution to get a closed form of the recurrence.

(c) Write the recurrence $S_k(n)$ for the number of times that `append` is called. Note that your result from part (b) may be helpful. Write 1-2 sentences explaining why the recurrence is correct. (You do not need to find a closed form of this recurrence.)

(2) Consider the following algorithm for computing $x^y$.

```python
def power(x, y):
    '''
    Given natural numbers x and y, returns x^y.
    '''
    if y == 0:
        return 1
    elif y % 2 == 0: # y is even
        t = power(x, y / 2)
        return t * t
    else:
        t = power(x, (y - 1) / 2)
        return t * t * x
```

(a) Write a recurrence, $T(y)$, for the number of times the multiplication operator is used. Write 1-2 sentences explaining why the recurrence is correct.

(b) Write a simpler version of the recurrence for the case where $y$ is a power of 2. Use repeated substitution to solve the recurrence in this case.