Problem 1. Complete the following recursive template function that implements ternary search.

template <typename T>
bool search(T x, T* A, int n)
{
    if (n < 10)
        for (int i = 0; i < n; ++i)
            if (A[i] == x)
                return _____;

    T m1 = A[n/3];
    T m2 = A[2*n/3];

    if (x == m1 _____ x == m2)
        return true;

    if (x < m1)
        return search(x, _____, n/3);

    if (x < m2)
        return search(x, _____, 2*n/3 - n/3);

    return search(x, _____, n-2*n/3);
}

Problem 2. Fill in the blanks below based on the search function in Problem 1.

The recurrence relation \( f(n) = \) \( \) with \( f(0) = \) describes the worst-case running time of ternarysearch.

Using repeated substitution once on the recurrence relation gives \( f(n) = \) .

A closed form for the worst-case running time of ternarysearch is \( \Theta(\) \).
Problem 3. Complete the following recursive template implementation of insertion sort.

```cpp
template <typename T>
void insertionsort(vector<T> &A)
{
    if (A.size() _____ 0)
        return;

    T last = A.back();

    A._____;
    insertionsort(______);

    int i = A.length()-1;
    while (i > 0 _____ A[i] < A[i-1])
    {
        T tmp = A[i];
        A[i] = A[i-1];
        A[i-1] = tmp;
    }
}
```

Problem 4. Fill in the blanks below based on the `insertionsort` function in Problem 3, where \( n \) is the length of the input vector \( A \).

A call to the `vector` class's `pop_back` method takes \( \Theta(_______) \) time.

The recurrence relation \( f(n) = \text{________________} \) with \( f(0) = \text{_______} \) describes the worst-case running time of `insertionsort`.

Using repeated substitution once on the recurrence relation gives \( f(n) = \text{________________} \).

A closed form for the worst-case running time of `insertionsort` is \( \Theta(\text{___________}) \).
Problem 5. Select the (worst-case) running time of each code snippet.

```java
for (int i = 0; i < n; ++i)
    search(A[i], A, n);
\[\Box \Theta(n) \quad \Box \Theta(n \log(n)) \quad \Box \Theta(n^2)\]

for (int i = 0; i < n; ++i)
    mergesort(A, n);
\[\Box \Theta(n) \quad \Box \Theta(n \log(n)) \quad \Box \Theta(n^2)\]

mergesort(A, n);
quicksort(A, n);
\[\Box \Theta(n) \quad \Box \Theta(n \log(n)) \quad \Box \Theta(n^2)\]

insertionsort(A, n);
quicksort(A, n);
\[\Box \Theta(n) \quad \Box \Theta(n \log(n)) \quad \Box \Theta(n^2)\]
```