1. Let \( T_{\text{bar}}(n) \) define the time complexity, as a function of \( n \), of executing \( \text{bar}(n) \), and let \( T_{\text{foo}}(n) \) define the time complexity, as a function of \( n \), of executing \( \text{foo}(n) \).

\[
\begin{align*}
\text{foo}(n : \text{integer}): \text{void} & \quad \text{bar}(k : \text{integer}): \text{void} \\
& \quad \begin{cases}
    \text{m: integer}; \\
    \text{m} := n \times n \times n; \\
    \text{bar} (\text{m}); \\
\end{cases} \\
& \quad \begin{cases}
    \text{if } (k \leq 1) \\
    \text{return}; \\
    \text{print("X");} \\
    \text{bar}(k / 2); \\
\end{cases}
\end{align*}
\]

Complete: \( T_{\text{bar}}(n) = O( \ ____________ ) \)

\( T_{\text{foo}}(n) = O( \ ____________ ) \)

Explain your answers.

2. For each of the following questions, briefly explain your answer.

a. If I prove that an algorithm takes \( O(n^2) \) worst-case time, is it possible that it takes \( O(n) \) on some inputs?

b. If I prove that an algorithm takes \( O(n^2) \) worst-case time, is it possible that it takes \( O(n) \) on all inputs?

c. If I prove that an algorithm takes \( \Theta(n^2) \) worst-case time, is it possible that it takes \( O(n) \) on some inputs?

d. If I prove that an algorithm takes \( \Theta(n^2) \) worst-case time, is it possible that it takes \( O(n) \) on all inputs?

3. Write (in pseudocode) a recursive function ‘MaxPair’ that gets an array \( a[] \) of integers and its size \( n \) (it is known that \( n>1 \)), and returns the maximal sum of two consecutive
elements in a[] (that is Max(a[j-1]+a[j] : 1 ≤ j ≤ n−1). You are not allowed to use loops in your solution.
What is the time and space complexity?

4. A ‘frame matrix’ is an N*N matrix in which all the values along the same frame are identical. For example, the matrix below is a 5*5 frame matrix.

```
7 7 7 7 7
7 14 14 14 7
7 14 -9 14 7
7 14 14 14 7
7 7 7 7 7
```

Suggest a data structure for storing a frame matrix, whose space complexity is O(N) (for an N*N frame matrix with N² elements). Using your suggested data structure, implement (write in pseudo-code) the following operations; each should have time complexity O(1):

- get(i, j) returns the value of the element whose location is (i,j).

- put(i, j, x) – set the value x at location (i,j) AND in all the locations in the frame to which (i,j) belongs, in a way that the resulting matrix is still a frame matrix.

5. t is a linked list. What is the result of executing rec_func(t) (defined below)? What is its time and space complexity? Explain briefly.

```c
rec_func1(t node_pointer, r node_pointer): node_pointer
{
    tail node_pointer;
    if (t = NULL) return r;
    tail := t.next;
    t.next := r;
    return rec_func1(tail, t);
}

rec_func(t node_pointer): node_pointer
{
    return rec_func1(t, NULL);
}
```