Instructions

The purpose of this take-home exam is to give me an idea of your level of comprehension of the pre-requisites for this class. The exam will not be graded. But you have to give honest effort, because based on your answers, I will be able to recommend you specific material to review before we jump into more advanced topics.

While taking the exam you may consult the CLRS textbook ("Introduction to Algorithms" by T. Cormen, C. Leiserson, R. Rivest, C. Stein, 3rd edition, 2009), but no other resources. You may NOT discuss this exam with anyone in or outside the class and should solve all problems by yourself.

Important: For each problem, please list how long it took you to solve it (including the time to review book chapters and/or course webpage).

1 Analysis of Algorithms

Analyze the runtime of the following algorithm. Show your work and justify your answer.

```python
Foo(n)
1  if (n < 2)
2    return 1
3  else
4    x = 0
5    for i = 1 to 3
6      x = x + Foo(n/2)
7    for i = 1 to n
8      for j = 1 to n
9        x = x + 1
10    return x
```

2 Solving Recurrences

Solve the following recurrences using any method you like, but you must justify your answers (i.e. show the proofs).

(a) \( T(n) = \begin{cases} 
2T(n/2) + \Theta(n) & \text{if } n > 1 \\
O(1) & \text{otherwise.} 
\end{cases} \)
(b) \( T(n) = \begin{cases} T(n/2) + \Theta(\log n) & \text{if } n > 1 \\ O(1) & \text{otherwise.} \end{cases} \)

(c) \( T(n) = \begin{cases} T(n-1) + \Theta(\log n) & \text{if } n > 1 \\ O(1) & \text{otherwise.} \end{cases} \)

(d) \( T(n) = \begin{cases} 2T(\sqrt{n}) + \Theta(1) & \text{if } n \geq 4 \\ O(1) & \text{otherwise.} \end{cases} \)

### 3 List ranking

Let \texttt{Node} be an abstract data type with the following specifications:

```
Node {
    Node next; // pointer to the next node in the list.
    int dist;  // distance value stored at each node.
}
```

The node data structure can be used to compose a singly linked list. Design an algorithm which takes a node object as the input (the head of the linked list) and updates the \texttt{dist} entry of every node of the linked list with the distance from the input node:

```java
rank(Node n) {

    return;
}
```

Thus, after the call to \texttt{rank(head)} the linked list should be updated in such a way, that

```java
head.dist = 0,
head.next.dist = 1,
head.next.next.dist = 2,
. . .
```