Problem 1 Suppose you are given an expression with $n$ paranthesis. Design an algorithm with running time $O(n)$ to check whether it is valid. Examples of valid expressions are (()) and ()(()(())). Examples of invalid expressions are ()()() and (())().
Problem 2  Function $T(n)$ is defined by the following recurrence relation.

$$T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
2T(\lfloor \sqrt{n} \rfloor) + 1 & \text{if } n > 1 
\end{cases}$$

Prove $T(n) = O(\log n)$. 
Problem 3 Function $T(n)$ is defined by the following recurrence relation.

$$T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
2T([\frac{n}{2}]) + n \log n & \text{if } n > 1 
\end{cases}$$

Prove $T(n) = O(n(\log n)^2)$ by the substitution method.
Problem 4 Solve the following recurrence relations using the Master Theorem, or just state that the Master Theorem does not apply.

(a) \( T(n) = 8T\left(\frac{n}{2}\right) + 10n^3 \log^2(n) \)

(b) \( T(n) = 5T\left(\frac{n}{3}\right) + O(n^{2.3}) \)

(c) \( T(n) = 4T(\sqrt{n}) + O(n^2) \)

(d) \( T(n) = 9T\left(\frac{n}{3}\right) + 100n^2 \)
Problem 5 Implement a queue by using two stacks.
Problem 6 Suppose you have a linked list of size \( n \). Give an \( O(n) \) algorithm to find out whether the linked list has a loop. Note that you cannot store the elements of the list in another array or anything like that. In other words, you can use extra \( O(1) \) space.

Note: We do not know the size of the linked list. The input is the pointer to the first element of the linked list.

Hint: Define two pointers and move one with speed 1 and another with speed 2!
Problem 7 Design an algorithm to find the lowest common ancestor of two given nodes in a balanced binary search tree. Your algorithm should runs in $O(\log n)$ where $n$ is the number of nodes.
Problem 8 Design an algorithm to find the next (e.g. in-order successor) node of a given node in a binary search tree.
Problem 9 Design an algorithm to construct a binary tree from given in-order and pre-order tree walks. (Note: This is not necessarily a binary search tree!)