1. A palindrome is a word, phrase or sequence that reads the same backward as forward, e.g., "Bob", or "A man, a plan, a canal: Panama". Using a stack, write a function that takes a string, which consists of only lowercase letters, and returns true if the string is a palindrome, false otherwise.

```cpp
#include <stack>

bool isPalindrome(const string& word) {
}
```

2. What does the following function compute?

```cpp
// b is a nonnegative integer
int mystery1(int a, int b) {
    if(b == 0) return 1;
    if (b % 2 == 0) return mystery1(a*a, b/2);
    return mystery1(a*a, b/2) * a;
}
```

3. What does the following function compute?

```cpp
// a and b are nonnegative integers
int mystery2(int a, int b) {
    if(b == 0) return 0;
    if (b % 2 == 0) return mystery2(a+a, b/2);
    return mystery2(a+a, b/2) + a;
}
```
4. Write a recursive function that takes an array of integers and its size as inputs, and prints its elements in reverse order, e.g., given the array $\{1, 5, 4, 7\}$, it should print $\{7, 4, 5, 1\}$.

```c
void printInReverseOrder(const int array[], int size) {
}
```

5. Draw a binary search tree with height 2 whose in-order traversal prints the nodes in the following order (there may be more than one such tree):

$$2, 3, 4, 5, 6, 8.$$ 

After that, state the outputs of post-order and pre-order traversals of this tree.
6. Consider a full binary tree, where each internal node has exactly two children. An internal node $N$ is called leftist iff both subtrees rooted at the children of $N$ are leftist and the smallest element in the left subtree of $N$ is bigger than the biggest element in the right subtree of $N$. A leaf node is called leftist by default. A full binary tree is leftist iff each of its nodes is leftist. Write a recursive function that takes a non-empty full binary tree, and returns true if the tree is leftist. You may need to define auxiliary functions.

```c
struct Node {
    int data;
    Node* left;
    Node* right;
};

bool isLeftist(const Node* node) { //node is non-empty
```