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) {
    stack<char> s;
    for(int i=0; i<word.size(); i++) s.push(word[i]);
    for(int i=0; i<word.size(); i++) {
        char ch = s.top();
        s.pop();
        if(word[i]!=ch) return false;
    }
    return true;
}
```
2. What does the following function compute?

```
// 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;
}
```

*a raised to power b.*
3. What does the following function compute?

```c
// 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;
}
```

\[ a \times b \]
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 $\{7, 4, 5, 1\}$.

```cpp
void printInReverseOrder(const int array[], int size) {
    if(size==0) return;
    printInReverseOrder(array+1, size-1);
    std::cout<< array[0] << " ";
}
```
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.

**Note:** I believe that these are all possible solutions. But, I may be missing some other valid solutions.
6. Consider a full binary tree, where each internal node has exactly two children. An internal node \( N \) is called leftist if 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.

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

int maxNode(const Node* node) {
    if(node == nullptr) return INT_MIN;
    int temp = node->data;
    int leftMax = maxNode(node->left);
    int rightMax = maxNode(node->right);
    if(leftMax > temp) temp = leftMax;
    if(rightMax > temp) temp = rightMax;
    return temp;
}

int minNode(const Node* node) {
    if(node == nullptr) return INT_MAX;
    int temp = node->data;
    int leftMin = minNode(node->left);
    int rightMin = minNode(node->right);
    if(leftMin < temp) temp = leftMin;
    if(rightMin < temp) temp = rightMin;
    return temp;
}

bool isLeftist(const Node* node) { //node is non-empty
    if(node->left == nullptr) return true; //if leaf
    if(isLeftist(node->left) && isLeftist(node->right))
        return minNode(node->left) >= maxNode(node->right);
    return false;
}
```

Note: INT_MIN and INT_MAX are defined in limits.h and denote the smallest and biggest integer values a C++ integer variable can take, respectively.