Q1

question 1:

The question is a nested for loop.

The first for loop for (int i = 1; i <= x; i = i * 2) indicates i will become 1,2,4,8,...,x.

The second for loop for (int j = 1; j <= x / i; ++j) indicates j will become 1,2,...,x/i.

Note that i follows the pattern 1,2,4,8,..., so j follows the pattern: from 1 to x, from 1 to x/2,..., from 1 to 4, from 1 to 2, from 1 to 1.

Totally, we need to consider x + x/2 + x/4 +...+ 2 + 1 elements

x + x/2 + x/4 + … + 2 + 1 = x/(1-½) = 2x

The Big-O notation of the function is O(N)

question 2:

It is a recursion function.

No matter b is even or odd, the recursive function calls itself 1 time

function(a,b) returns (function(a,b/2))^2 or a*(function(a,b/2))^2, both of them consider 1 time only.

Consider function(a,b), it consider function(a,b/2), function(a,b/4),…, function(a,2), function(a,1), function(a,0),

There are total log(b) levels.

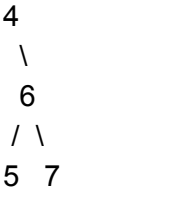
The Big-O notation of the function is O(log(N)).

Q2

In this question, I use examples to help illustrating the answers.

(1)

Example



Inorder traversal(left->root->right) of the tree: { 1, 2, 3, 4, 5, 6, 7 }

Postorder traversal(left->right->root) of the tree: { 1, 3, 2, 5, 7, 6, 4 }

The answer is yes.

First, we obtain the last item by considering the postorder transversal. The last item should be the root of that binary search tree. i.e. 4. So we insert 4 into the new tree.

Second, we can get the left subtree and right subtree by considering the left part and right part of inorder traversal. i.e. 1,2,3 is in the left subtree. 5,6,7 is in the right subtree.

We repeat this recursively. Therefore,

left subtree:

Inorder traversal(left->root->right) of the tree: { 1, 2, 3 }

Postorder traversal(left->right->root) of the tree: { 1, 3, 2 }

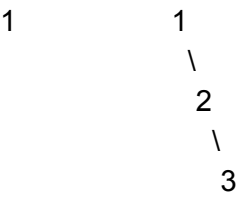
right subtree:

Inorder traversal(left->root->right) of the tree: { 5, 6, 7 }

Postorder traversal(left->right->root) of the tree: { 5, 7, 6 }

We can always construct the unique binary tree.

(2)
Example



Preorder traversal(root->left->right) of the tree: { 1, 2, 3 }
Postorder traversal(left->right->root) of the tree: { 3, 2, 1}

The answer is no. There are ambiguity.

In here, we cannot construct an unique binary tree.

Both examples of trees satisfy that postorder and preorder. It is because th skewed tree cause the ambigrity.