Questions:

1. Consider stack $S$, which is initialized with values 17, 330, and 4, in that order (with 17 at the top of stack). What are the contents of $S$ after the execution of the following code, where `S.push()` pushes a new value onto the top of the stack, and `S.pop()` removes the item from the top of the stack and returns its value:

```java
int x = 7;
S.push(32);

while(x != 20)
{
    x = S.pop();
    x--;
    S.push(x);
}
```
S.push(x + S.pop() + S.pop());

2. Why is it inefficient to use a recursive function to compute Fibonacci numbers? In what other conditions may it be inappropriate to use a recursive solution?

Recall that \( \text{fibonacci}(n) = \text{fibonacci}(n - 1) + \text{fibonacci}(n - 2) \)

3. What are some of the most common applications of trees?

4. If you have a balanced tree in which every node has a given, fixed number of children, is it necessary to explicitly create a new structure in order to represent the tree in memory, or is it possible to represent the tree by using only built-in types (assume that you are using C++)? Why or why not?

5. Consider the case described in the above example, except in this case a node may have a variable number of children. Is it necessary to explicitly define a new structure to represent this tree? Why or why not?

6. What are the advantages of using a tree structure to store data instead of a linked list? What are the disadvantages?

7. Consider the tree pictured below, is it a valid binary search tree? Why or why not?

```
  12
 /   \
6    17
 / \
2   8 14
 / \
1  4
 / \
3 7
```
8. Why is deletion problematic when dealing with a properly sorted binary search tree? What happens in the case when a leaf node is deleted? A node with a single child? A node with two children?

9. What are the fundamental properties associated with a B-tree of order M? In what kind of situation would it be desirable to use a B-tree to store data?

10. What is the difference between a tree and a graph? Are all trees graphs? Are all graphs trees? If a graph is acyclic, is it necessarily equivalent to a tree?

11. (15 Bonus Points) Suppose that graph G has the following adjacency lists:

   1 – (2; 3; 4)
   2 – (1; 3; 4)
   3 – (1; 2; 4)
   4 – (1; 2; 3; 6)
   5 – (6; 7; 8)
   6 – (4; 5; 7)
   7 – (5; 6; 8)
   8 – (5; 7)

(1) (5 Bonus Points) Draw G.

(2) (5 Bonus Points) Give the sequence of vertices visited using depth-first search starting at vertex 1.

(3) (5 Bonus Points) Give the sequence of vertices visited using breadth-first search starting at vertex 1.