There are five problems on this exam spread over five pages. Please do not hesitate to ask for help if you are not sure what you have been asked to do.

**Problem 1 (18 points)**

Determine the (worst-case) run-time requirements of the following algorithms. In each case, assume that the data-structure that is subject to the operation has $n$ values stored in it. Justify your answer using a short and concise sentence. Six items, three points for each item.

1. A linked-list of ordered values:
   (a) insert
   (b) delete
   (c) lookup

2. An array-based list of unordered values:
   (a) insert
   (b) delete
   (c) lookup

3. An array-based list of ordered values:
   (a) insert
   (b) delete
   (c) lookup

4. Insertion-Sort.

5. Verifying whether the values stored in a linked-list are sorted in increasing order of values.

6. Reversing the values in an array by swapping the element at $i, 0 \leq i < n/2$ with the element at $n - i - 1$. 
Problem 2 (25 points)

Using the primitive functions (null, is_null, atom, eq, car, cdr, cons) and recursion, write a function to find the depth of a recursive-list. Here are a few examples.

1. The depth of () is 1.
2. The depth of ( () ) is 2. This is because there is a “()” inside of another one.
3. The depth of ( ( () ) ( ) ) is 3. This is because there is a “()” inside of another “()” which is inside of another “()”. Note that there are two “()” at depth 3 and one at depth 2.
4. The depth of ( ( () ) ( ( () ) ( ) ) ) is 5.

int depth( list p )

// pre-conditions:
// p is a recursive-list that is neither an atom nor does it contain any atoms.
// Return the depth of ‘p’
Problem 3 (10 points)

Write a C++ implementation for the following function. Note that “set1” and “set2” represent values in the range [0, 64).

```cpp
int numElementsInCommon(unsigned long set1, unsigned long set2);
// This function finds and returns the number of elements that the sets have in common.
// For example, if the following is the right-most 8 bits of "set1" and "set2"
// (a space is used to make the numbers more easily readable):
// 8 bits of set1: 0010 1110
// 8 bits of set2: 0010 0100
// This function should return 2. This is because 3 and 6 are members of both sets.
```
Problem 4 (25 points)

Given two binary search trees, determine whether they share at least one value.

```cpp
bool twoTheSame( TreeNode *p, TreeNode *q )
// p and q are pointers to two binary search trees. The function
// returns true if p and q have at least one value in common. Otherwise,
// it returns false. To solve this problem, it is fine to write
// a helper function.
```

Here is a definition for a binary search tree node.

```cpp
class TreeNode {
public:
    TreeNode(): left(0), right(0), data(0) {}  
    TreeNode( int n ): left(0), right(0), data(n) {}  

    TreeNode *leftSubtree() { return left; }
    TreeNode *rightSubtree() { return right; }

    void leftSubtree( TreeNode *left ) { this->left = left; }
    void rightSubtree(TreeNode *right) { this->right = right; }

    int& value() { return data; }
private:
    TreeNode *left, *right;
    int data;
};
```
Problem 5 (25 points)

Let $T_1$ and $T_2$ be two binary search-trees. For this problem, we are interested in the structure of the trees and not the values that they hold. $T_1$ as a substructure includes $T_2$ if the structure of $T_2$ is embedded in $T_1$ at some node. For example, in the following figures, the tree at the bottom, $T_2$, is a substructure of $T_1$, the tree above it. The nodes in $T_1$ that have small squares in them show the structure of $T_2$. You are to write a Boolean function that returns true, if the second argument is a substructure of the first argument. Otherwise, the function should return false. Use the definition of TreeNode from the previous problem.

bool isSubstructureOf( TreeNode *t1, TreeNode *t2 )
// Is t2 a substructure of t1?

$T_1$ (top tree) and $T_2$ (bottom tree)