A binary search tree is either empty, or it consists of a node with two binary search trees as subtrees. Each node holds an integer. The elements in a binary search tree are arranged so that smaller elements appear in the left subtree of a node and larger elements appear in the right subtree.

Let atom empty represent an empty binary search tree and let a term node(K, S, T) represent a tree with an integer value K at the root, left subtree S and right subtree T.

1) Define a relation \texttt{min} to extract the minimum element in a tree. The two arguments of \texttt{min} are the tree and an integer representing the smallest element in the tree.

2) Define a relation \texttt{max} to extract the maximum element in a tree. The two arguments of \texttt{max} are the tree and an integer representing the largest element in the tree.

3) Define a relation \texttt{height} to calculate the height of a tree. The two arguments of \texttt{height} are the tree and an integer representing the height of the tree.

4) Define a relation \texttt{nodecount} to calculate the number of nodes in a tree. The two arguments of \texttt{nodecount} are the tree and an integer representing the number of nodes in the tree.

5) Define a relation \texttt{member} to test whether an integer appears at some node in a tree. The two arguments of \texttt{member} are an integer and a tree.

6) Define a relation \texttt{insert} to insert an element \(e\) into a tree \(T\). The three arguments of \texttt{insert} are an integer, the original tree and the tree after the insertion.

7) Define a relation \texttt{delete} to delete an element \(e\) from the tree \(T\). The three arguments of \texttt{delete} are an integer, the original tree and the tree after the deletion.

8) Define a relation \texttt{inorder} to perform an inorder traversal of a tree. The two arguments of \texttt{inorder} are the tree and a list whose elements are the elements of the tree in sorted order.

To make your lives easier, assume that the max of an empty tree is 0 and the min of an empty tree is 1000. As such, valid tree elements are between 1 and 999.
Submission:

To facilitate automatic testing of your program, you should name the file containing your program “a3.pl”. Only the code needs to be submitted electronically. Use the following command:

```
submit -c csc324h -a A3 a3.pl
```

Your code should be well documented with comments. In addition, you need to submit a test report that shows the tests you performed on your program. A rationale should be provided for the test cases and these should be exhaustive. A hardcopy of the code and test report must be submitted in tutorial. If desired, you may create and use helper predicates. Keep checking the course web page to become aware of any late-breaking information about the assignment as it becomes available.

Tentative Marking Scheme

50% Automated Testing
30% Thoroughness of test report
10% Comments
10% Coding Style