ECE 250 *Algorithms and Data Structure*
Department of Electrical and Computer Engineering
University of Waterloo
Please send any comments or criticisms to dwharder@alumni.uwaterloo.ca
with the subject ECE 250 Questions 6.1.
Assistances and comments will be acknowledged.

6.1*a* Insert the following 15 randomly generated objects into a binary search tree in the order they are listed.

34, 15, 65, 62, 69, 42, 40, 80, 50, 59, 23, 46, 57, 3, 29

6.1*b* What is the average depth of a node in this tree? How does this compare to the average depth of a perfect binary tree of height 3?

6.1*c* Give two integers that could be inserted into this tree that would increase the height of this tree.

6.1*d* Remove the root node four times by copying up the smallest element of the right sub-tree.

6.1*e* In order to find the *k* item in a binary tree in O(*h*) time, it is necessary that each node stores the number of nodes in the sub-tree rooted at that node. Consequently, the node class will require an additional member variable:

```
template <typename Type>
class Binary_search_node:public Binary_node<Type> {
    using Binary_node<Type>::element;
    using Binary_node<Type>::left_tree;
    using Binary_node<Type>::right_tree;
    int tree_size;
```

We will have to override the `int Binary_node::size() const` function to return this variable:

```
template <typename Type>
int Binary_search_node::size() const {
    return tree_size;
}
```

Update the constructor, insertion, and erase functions to accommodate this new member variable.

6.1*f* Create the binary search tree by inserting the elements

2, 1, 4, 3, 6, 5, 8, 7, 10, 9, …

and so on, in that order. What is a good description on the run-time of the find member function on the resulting binary search tree?