ECE 250
Data Structures and Algorithms
QUIZ 3

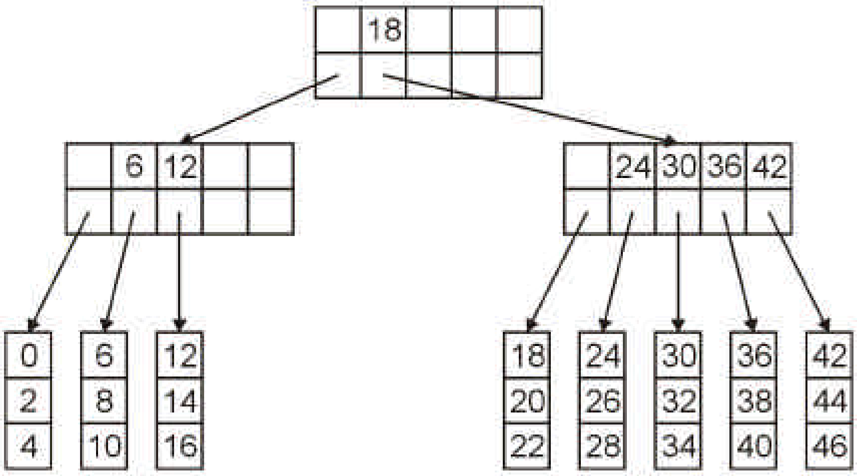
2006-11-05

The quiz is out of 20 marks.
No questions, no aides.
If you are unsure about a question, write down your assumptions and continue.
This examination has two pages of questions.
If you run out of room, use the reverse of this page.

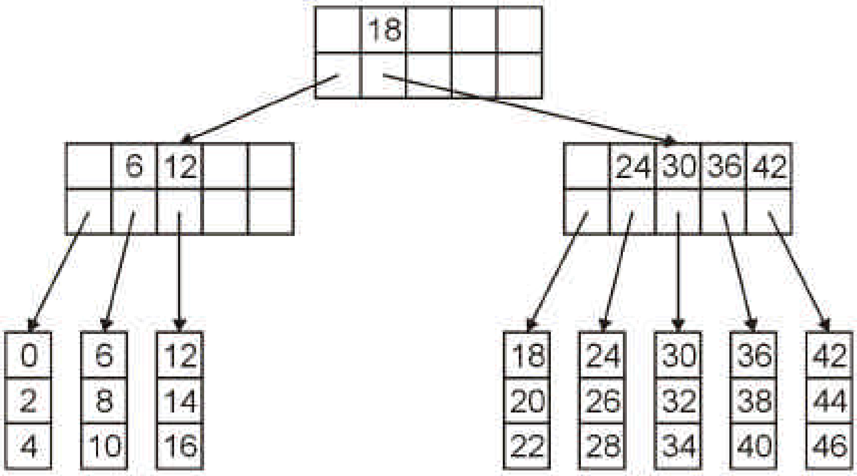
Surname, Given Name				**Student ID**	
1.	**2.**	**3.**	**4.**	**5.**	**B.**

Sign here to indicate that you have read the above instructions.

1. **[3]** Insert 11 into the following B-tree with M = 5 and L = 3. You need only redraw those nodes which have changed.

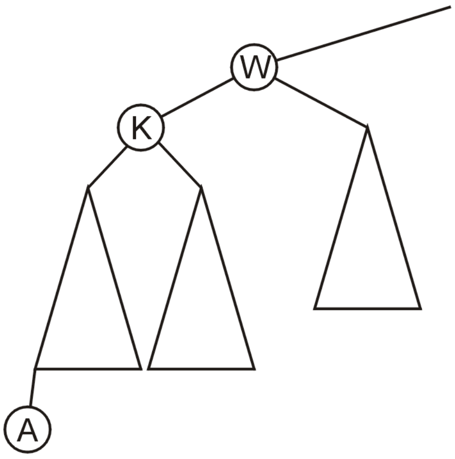


Insert 33 into the following B-tree with M = 5 and L = 3. You need only draw those nodes which have changed. Do not transfer leaf nodes to the 1st sub-tree of the root node.

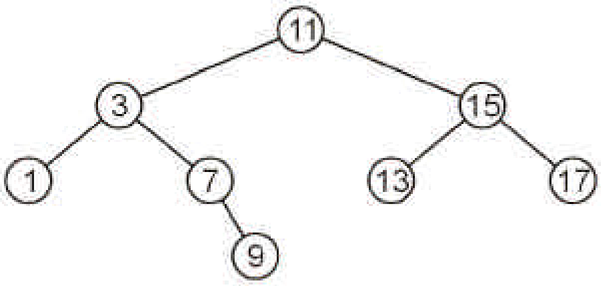


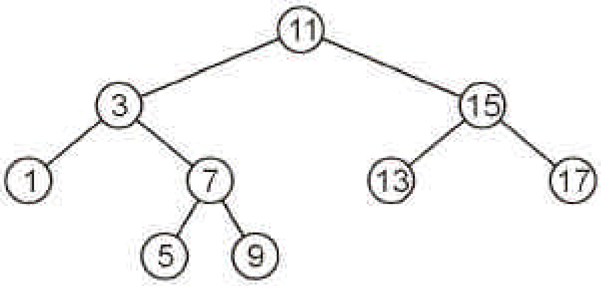
2. **[4]** What are the maximum and minimum number of records which can be stored in a B tree of height 5 with M = 512 and L = 32. Write your answer as a product of integers but **do not** calculate any of the products.

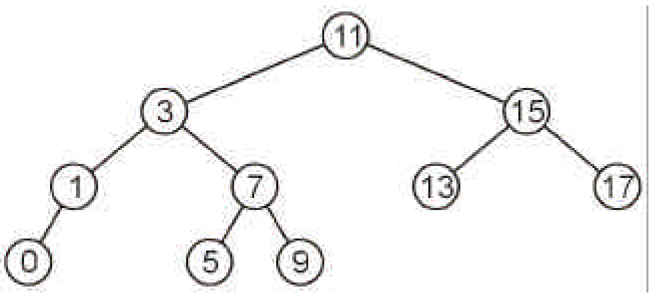
3. **[3]** Use the following diagram to justify that if the insertion of A (A < K < W) causes an AVL-imbalance at node W that, once the appropriate rotation is made, no further AVL imbalances can occur for any of the ancestors of W. Assume that the height of the sub-tree with root W before the insertion of A was *h*.



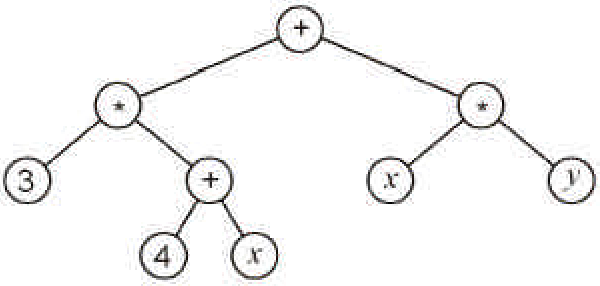
4. **[6]** Insert 10 into each of the following three AVL trees, performing whatever rotations are necessary to restore AVL balance.







5. **[4]** As described in class, the tree



represents the expression 3(4 + *x*) + *xy*. Assume each leaf node is either a numeric value or a variable name. Assume each internal node is either an addition or a multiplication operator.

```
template <typename Object>
class ExpressionTree {
        private:
                ExpressionTree * left_tree;
                ExpressionTree * right_tree;
                // other member variables

        public:
                bool is_leaf();         // true if number or variable
                bool is_numeric ();     // true if number and leaf
                bool is_variable ();    // true if variable and leaf
                bool is_product();      // true if an operator is product
                                        // false if it is a sum
                int get_numeric();      // returns the numeric value of a
                                        // numeric leaf node
                void traversal( const ExpressionTree * & to_this );
                // other member functions
};
```

The implementation of the `traversal` function is:

```
void traversal( const ExpressionTree * & to_this ) {
        if ( is_leaf() ) {
                return;
        }

        left_tree -> traversal( left_tree );
        right_tree -> traversal( right_tree );

        if ( is_product() ) {
                if ( left_tree -> is_numeric() &&
                     left_tree -> get_numeric() == 1 ) {
                         to_this = right_tree;
                         return;
                }

                if ( right_tree -> is_numeric() &&
                     right_tree -> get_numeric() == 1 ) {
                         to_this = left_tree;
                         return;
                }
        } else {
                if ( left_tree -> is_numeric() &&
                     left_tree -> get_numeric() == 0 ) {
                         to_this = right_tree;
                         return;
                }

                if ( right_tree -> is_numeric() &&
                     right_tree -> get_numeric() == 0 ) {
                         to_this = left_tree;
                         return;
                }
        }
}
```

Describe, in words, what this function does. You can give an example, if you wish.