ECE 250
Data Structures and Algorithms
MIDTERM EXAMINATION

2009-02-27/17:30-18:45
MC 1085 20100000 to 20259999 72 students
MC 2054 20260000 to 20268499 48 students
MC 4064 20268500 to 20272999 18 students
MC 4045 20273000 to 20289999 58 students

Instructions:

No aides.
Turn off all electronic media and store them under your desk.
If there is insufficient room, use the back of the previous page.
You may ask only one question during the examination:

"May I go to the washroom?"

If you think a question is ambiguous, write down your assumptions and continue.
Do not leave during the first 30 minutes of the examination.
Do not leave during the last 15 minutes of the examination.
Do not stand up until all exams have been picked up.
Bonus of 1 for obeying all these instructions.
You must stop writing when you are told that the examination is finished. No exceptions, not even for your name. (-5 mark penalty.)

Attention: The questions are in the order of the course material, not in order of difficulty.
I have read and understood all of these instructions and will accept a grade of 0 if I fail to follow them:

Surname: _____

Given Name(s): _____

UW Student ID Number: _____

Signature: _____

Relationships

1. **[4]** Fill in the blanks with the most appropriate word or phrase.

C# classes and directories in Unix are all related to a single object, either the `Object` class or a root directory. Therefore, these may be described using a _____ relationship. Two C++ classes or two Windows directories may not share a common ancestor (base class or drive) and therefore may only be described using a _____ relationship.

Given two functions f(n) and g(n), the relationship f(n) = Q(g(n)) forms an _____ relation which allows us to group related functions. The relationship where a person A is related to person B if they share the same mother _____ (use *does* or *does not*) satisfy this same type of relationship.

Runtime Analysis

2. **[4]** Fill in the Table 1 to include four more functions $f_k(n)$ for k = 2, 3, 4, 5 so that the relationship $f_k(n) = \mathbf{o}(f_{k+1}(n))$ holds.

Table 2. Functions.

$f_1(n) = n$	$f_2(n)$	$f_3(n)$	$f_4(n)$	$f_5(n)$	$f_6(n) = n^2 \ln(n)$

3. **[3]** Determine, using limits, which Landau symbol, **o, O**, Q, w, or W best describes the relationship between $n^2 \ln(n)$ and n^3. State your answer in the form $n^2 \ln(n) = \mathbf{?}(n^3)$. You must use l'Hopital's rule where necessary.

4. **[2]** While lg(n) = Q($\log_{10}(n)$), it is not true that 2^n = Q(10^n). Show this. Which Landau symbol would you use to describe the relationship $2^n = \mathbf{?}(10^n)$?

5. **[5]** Associate each of the five run times

a. $T(n) = \Theta(n)$ _____
b. $T(n) = \mathbf{O}(n^2)$ _____
c. $T(n) = \Theta(n^2)$ _____
d. $T(n) = T(n/2) + \Theta(1)$ _____
e. $T(n) = T(n - 1) + \Theta(n)$ _____

with the five functions `f1`, `f2`, `f3`, `f4`, and `f5` listed here:

```
int f1( int n ) {
  int sum = 0;
  for ( int i = 0; i < n; ++i ) {
    for ( int j = std::min( i, n - i ); j < std::max( i, n - i ); j++ ) {
      ++sum;
    }
  }
  return sum;
}

int f2( int n ) {
  int sum = 0;

  for ( int i = 0; i < n; ++i ) {
    ++sum;
  }

  return sum + f2( n – 1 );
}

int f3( int n ) {
  int sum = 0;
  for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < n; ++j ) {
      if ( (i + j) * 123456789 % 1035 == 0 ) {
        break;
      } else {
        ++sum;
      }
    }
  }
  return sum;
}

int f4( int n ) {
  return 1 + f4( n/2 );
}

int f5( int n ) {
  int sum = 0;
  for ( int i = 0; i < n; ++i ) {
    for ( int j = i – 1; j <= i + 1; ++j ) {
      ++sum;
    }
  }
  return sum;
}
```

6. **[4]** Using a proof by induction, show that $\sum_{k=0}^{n} k2^k = 2\left(n2^n - 2^n + 1\right)$.

Arrays and Linked Lists

For your information, these are the declarations of the **`Single_list`** and **`Single_node`** classes from Project 1.

```
template <typename Object>
class Single_list {
    private:
            Single_node<Object> * list_head;
            Single_node<Object> * list_tail;
            int count;

    public:
            Single_list();
            Single_list( const Single_list & list );
            ~Single_list();

            Single_list & operator = ( const Single_list & rhs );

            int size() const;
            bool empty() const;
            Object front() const;
            Object back() const;
            Single_node<Object> * head() const;
            Single_node<Object> * tail() const;

            bool member( const Object & obj ) const;

            void push_front( const Object & obj );
            void push_back( const Object & obj );
            Object pop_front();
            bool remove( const Object & obj );
};

template <typename Object>
class Single_node {
    private:
            Object       element;
            Single_node * next_node;

    public:
            Single_node( const Object & e = Object(), Single_node * n = 0 );
            Object retrieve() const;
            Single_node *next() const;

            friend class Single_list<Object>;
};
```

7. **[4]** Implement a function `range_find( Object a, Object b ) const` which returns the smallest object x in the linked list which satisfy the mathematical condition $a = x = b$. If no such object is found, return b. If $a > b$ then throw an `invalid_range` exception. The run time must be $\Theta(n)$.

```
template <typename Object>
Object Single_list<Object>::range_find( Object a, Object b )
const {




}
```

Stacks and Queues

8. **[4]** Which of the following reverse-Polish expressions are valid; that is, they may be evaluated using a stack and the result is a single number.

a. 3 5 + 2 + + ×
b. 4 3 + 7 × 5 + 8 +
c. 5 + 9 2 5 + ×
d. 3 6 2 + 7 5 + ×

Tree Definitions

9. **[2]** How many paths of length 2 and how many paths of length 3 are in the binary search tree shown in Figure 9?

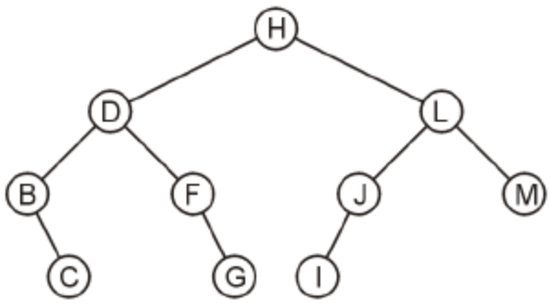

Figure 9. A binary search tree.

10. **[4]** Describe with pictures or words how you would find the next-largest element in a binary search tree given a specific node with a given value. Under what conditions does a node not have a next-largest element?

Tree Traversals

11. **[4]** For each of the following, indicate whether the function prints the nodes in a pre-order depth-first traversal order, post-order depth-first traversal order, or breadth-first traversal order. The stacks and queues work as expected from class.

```
template<typename Object>                    ____________________________________
void Binary_tree<Object>::f() {
    stack< Binary_tree_node<Object> * > stk;

    stk.push( root );

    while ( !stk.empty() ) {
        Binary_tree_node<Object> *ptr = stk.pop();
        if ( right() != 0 ) {
             stk.push( right() );
        }
        if ( left() != 0 ) {
             stk.push( left () );
        }
        std::cout << ptr->retrieve() << std::endl;
    }
}
```

```
template<typename Object>                    ____________________________________
void Binary_tree_node<Object>::g() {
    if ( this == 0 ) {
        return;
    }

    std::cout << retrieve() << std::endl;
    left()->g();
    right()->g();
}
```

```
template<typename Object>                    ____________________________________
void Binary_tree_node<Object>::h() {
    if ( left() != 0 ) {
        left()->h();
    }
    if ( right() != 0 ) {
        right()->h();
    }
    std::cout << retrieve() << std::endl;
}
```

```
template<typename Object>                    ____________________________________
void Binary_tree<Object>::k() {
    queue< Binary_tree_node<Object> * > que;

    que.enqueue( root );

    while ( !que.empty() ) {
        Binary_tree_node<Object> *ptr = que.dequeue();
        if ( left() != 0 ) {
             que.enqueue( left() );
        }
        if ( right() != 0 ) {
             que.enqueue( right() );
        }
        std::cout << ptr->retrieve() << std::endl;
    }
}
```

Binary Search Trees

12. **[4]** Without any balancing, insert the values 8, 3, 6, 9, 1, 4, 7, 2, 5 into an initially-empty binary search tree. The elements **must** be inserted in the given order.

13. **[4]** There are three cases that must be considered when removing a node from a binary search tree: the node being removed may be full, it may have one child, or it may be a leaf node. Are all these cases applicable when removing either the minimum or maximum elements in a binary search tree?

14. **[4]** Implement a function `void remove_min(...)` which removes the minimum element from a binary search tree and correctly updates the other nodes in the tree. The removal cannot change the order; *i.e.*, it must continue to be a binary search tree.

```
template<typename Object>
void Binary_search_tree<Object>::remove_min() {
    if ( empty() ) {
        throw underflow();
    }
    root()->remove_min( root_node );
}

template<typename Object>
void Binary_search_tree_node<Object>::remove_min(
        Binary_search_tree_node<Object> *&ptr_to_this
) {









}
```

15a. **[3]** Figure 15 shows an AVL tree. Give an example of a number which, if inserted, causes no rotations, an example of a number which causes a single rotation, and an example of a number which causes a double rotation.

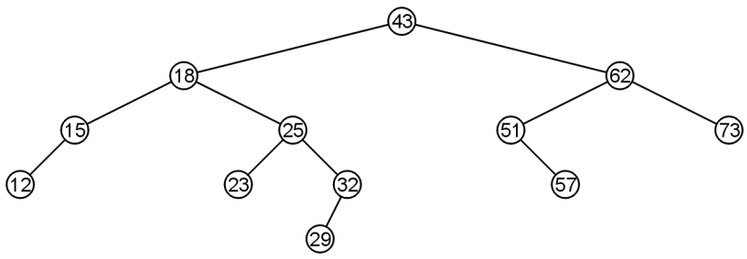

Figure 15. An AVL tree.

No rotations: _____ A single rotation: _____ A double rotation: _____

15b. **[2]** Given the AVL tree in Figure 15, give two numbers which, if inserted, cause Node 25 to become unbalanced and which require single rotation at Node 25 to correct.

16. **[2]** Given the AVL tree in Figure 16, insert the value 22 and, if necessary, rebalance the tree. Only redraw the subtree including the node which became imbalanced.

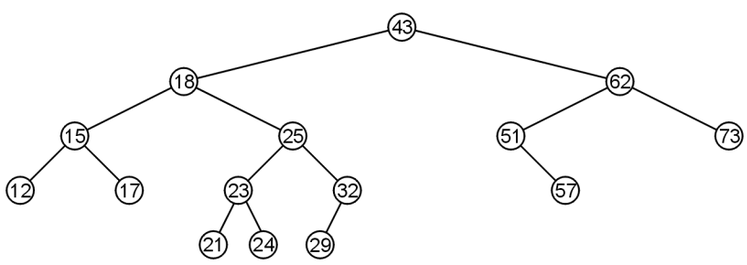

Figure 16. An AVL tree.

17. **[3]** In class, it was indicated that an insertion could be fixed with one AVL rotation (either single or double). If we delete 73 from the AVL tree in Figure 18, show that we require two rotations to correct the AVL imbalance.

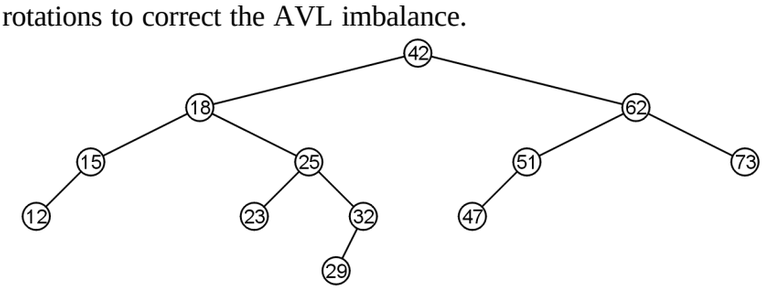

Figure 18. An AVL tree.

Unix

18. **[3]** Correct the following Unix commands so that they do what you would expect them to do:

```
{eceunix:1} cd..
{eceunix:2} tar –cvf *.cpp *.h
{eceunix:3} g++ *.h Single_list_int_driver.cpp
```