ECE 250
Data Structures and Algorithms
MID-TERM EXAMINATION

2009-10-29/08:30-9:50
RCH 105, RCH 110

Instructions:

There are 70 marks and the examination will be marked out of 65.
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 examination period.
Do not stand up until all exams have been picked up.

Attention:

The questions are in the order of the course material, not in order of difficulty.

Surname, Given Name: _____

UW Student ID Number: | | | | | | | | |

UW User ID: | | | | | | | | |

Signature: _____

Relationships

1. **[1]** Does $f(n) = \mathbf{o}(g(n))$ describe a linear order or an equivalence relation?

2. **[3]** What are the three relations other than those listed in Question 1 which we saw in this class?

Runtime Analysis

3. **[4]** List five functions ordered f_1, ..., f_5 so that $f_k = \mathbf{o}(f_{k+1})$ and $f_1 = \omega(\ln(n))$ and $f_5 = o(n^2)$. Place your answers in Table 3.

Table 3.

f_1	f_2	f_3	f_4	f_5

4. **[3]** Determine, using limits, which Landau symbol, **o, O**, Θ, ω, or Ω best describes the relationship between $\ln(n)$ and e^n. State your answer in the form $\ln(n) = \mathbf{?}(e^n)$. You must use l'Hopital's rule where necessary.

5. **[4]** What are the run times, using big-Θ notation of the following two pieces of code? Show your work.

```
for ( int i = 0; i < n; ++i ) {
     for ( int j = 0; j < n/2; ++j ) {
          sum += 1;
     }
}

for ( int i = 0; i < n; ++i ) {
     for ( int j = 0; j < n/2; ++j ) {
          sum += i;
     }

     for ( int k = n/2; k < n; ++k ) {
          sum += i;
     }
}

for ( int i = 0; i < n; ++i ) {
     for ( int j = 0; j < i; ++j ) {
          for ( int k = 0; k < j/i; ++k ) {
               sum += i;
          }
     }
}
```

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

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. **[9]** Implement a member function **`remove_max`** which traverses a linked list once and then removes the **last** occurrence of the largest object in the linked list (you must assume that there may be duplicate entries). You may use any member function which you implemented in Project 1. You can only use **O**(1) additional memory over and above the memory in the original linked list. Throw an underflow if the linked list is empty.

```
template <typename Object>
void Single_list<Object>::remove_max() {
```

```
}
```

Stacks and Queues

8. **[3]** Suppose we are using a stack to match opening and closing delimiters (parentheses, brackets, and braces, *i.e.*, (), [], and {}) in C++ code. There are a number of situations which may cause an error in parenthesis matching. Complete the sentences:

The stack is empty but

The stack is not empty but

A closing delimiter is appears in the code but

9. **[1]** Evaluate the following expression which uses reverse-Polish notation:

1 2 3 + * 4 5 * 6 + +

10. **[3]** Suppose a queue, as implemented in Project 2, is stored in an cyclic array of size *n* which is full. Describe, using words and images, how you would double the size of the array and redistribute the *n* objects currently in the queue. Indicate the front and back of the queue in any diagrams.

Tree Definitions

11. **[3]** Give the definitions of the following terms in the context of a general tree:

a. A path of length *n*

b. The depth of a node

c. Descendants of a node

12. **[2]** A general tree is used to store a _____ ordering while a binary search tree is used to store a _____ ordering.

13. **[3]** What is the average depth of a node in a perfect binary tree of height *h* and justify your answer. Give the exact formula and the asymptotic average depth (using Theta notation). Hint: Look up, look way up!

Tree Traversals

14. **[4]** Perform an in-order and a breadth-first traversal of the binary tree shown in Figure 14 and place your answer in Tables 14a and 14b, respectively.

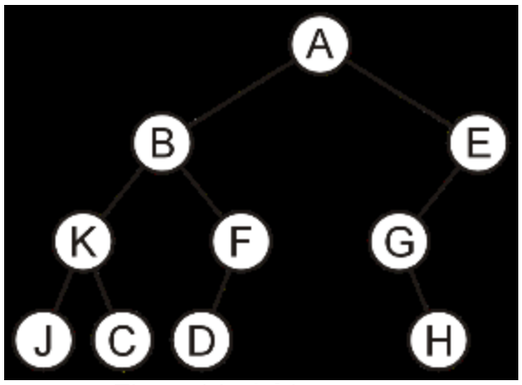

Figure 14. A binary tree.

Table 14a. In-order depth-first traversal.

Table 14b. Breadth-first traversal.

15. **[2]** In class, it was described how a queue may be used for a breadth first traversal. List the objects in the queue during a breadth-first traversal of the binary tree in Figure 14 when F is at the front of the queue but before any operations on the node containing F have been performed. Place your answer in Table 15.

Table 15. A queue.

Binary Search Trees

16. **[4]** Without any balancing, insert the elements 6, 8, 9, 3, 1, 4, 2, 0, 5, 7 into an initially-empty binary search tree. The elements **must** be inserted in the given order.

17. **[3]** Suppose we have a binary search tree and while performing a traversal, we insert the objects into a new binary search tree as we visit each node. Which of the four traversals (breadth-first; and pre-, in-, and post-order depth-first traversals) will produce the identical binary search tree (same root, children, *etc.*)?

18. **[7]** Implement a function **`remove_front`** which removes the least object in a binary search tree. You may assume that the `BST_node` class has two relevant member variables `left_tree` and `right_tree`. Remember to `delete this`.

```
template <typename Object>
void BST<Object>::remove_front() {
    if ( empty ) { throw underflow(); }
    root_node->remove_front( root_node );
}

template <typename Object>
void BST_node<Object>::remove_front( BST_node * &ptr_to_this ) {









}
```

AVL Trees

19. **[4]** In each of the following three parts, you are required to start with the AVL tree shown in Figure 19. Do not use the tree which is modified from the previous part. You are only required to draw the descendants of any node where an AVL imbalance occurred.

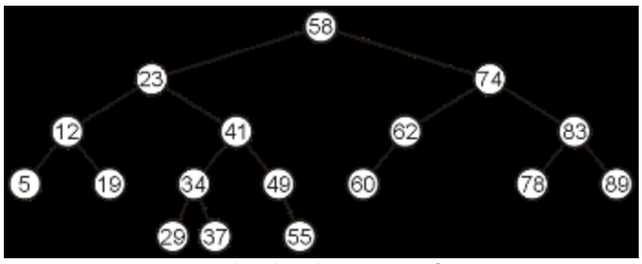

Figure 19. An AVL search tree.

a. Insert 59 into the AVL tree in Figure 19 and balance if necessary.

b. Insert 53 into the AVL tree in Figure 19 and balance if necessary.

c. Insert 25 into the AVL tree in Figure 19 and balance if necessary.

Unix

20. **[3]** List six Unix commands.