ECE 250
Data Structures and Algorithms
MID-TERM EXAMINATION

2008-2-13/5:30-7:00

The examination is out of 64 marks.

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.
I will, for a bonus mark of 1, not leave the last question blank.

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:

Name: ____

Signature: ____

Relationships

1. **[4]** For each of the following, indicate which relationship may be used in the description of the following concepts:

   f(x) = Q(g(x))

   A binary search tree

   A general tree

   A binary search on an array

Runtime Analysis

2. **[4]** Place the following six functions into Table 1 in the order $f_1, \ldots, f_6$ so that f_k = **O**(f_{k+1}). (There may be multiple answers if f_k = Q(f_{k+1})). Be sure to place them in the correct (**not reversed**) order. -1 for each inversion for a minimum score of 0.

$2n\ln(n)$	$17\,n^2 \ln(n)$	$5n + n\ln(n)$
9	$\log_{10}(n)$	$18 + 9\,n^2 \lg(n)$

Table 1.

3. **[4]** Determine, using limits, which Landau symbol, **o, O**, Q, w, or W best describes the relationship between $n^{1/2} \ln(n)$ and n. State your answer in the form $n^{1/2} \ln(n)$ = **?**(n). You must use l'Hopital's rule where necessary.

4. **[2]** Which Landau symbol best describes the relationship between $n \lg(n)$ and $n \ln(n^2)$

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

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

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

6. **[4]** Using a proof by induction, show that $\sum_{i=1}^{n} i(i+1) = \frac{2n+3n^2+n^3}{3}$.

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. **[6]** Implement a member `reverse` which reverses the elements in a linked list. You may use any member functions which you implemented in Project 1. You can only use **O**(1) additional memory over and above the memory in the original linked list and the memory in the new linked list. Hint: consider using

```
    Single_node<Object> ptr = head();
    list_head = 0;
    list_tail = 0;
    count = 0;
```

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
















}
```

8. **[4]** Implement a routine which pops the second element from the singly linked list you implemented in Project 1. It should throw an underflow exception if the linked list has fewer than two elements. The head should remain unchanged. You can use any member functions you implemented in Project 1.

```
template <typename Object>
Object Single_list<Object>::pop_second_from_front() {
















}
```

Stacks and Queues

9. **[3]** Given the following xhtml, what is the state of a stack which stores opening xml tags which are to be matched with appearing closing tags by the time we reach the final character **n**?

```
<xhtml>
<body>
<p>1b. If lg(<i>n</i>) = log<sub>2</sub>(<i>n</i>) = <i>x</i> then what
is log<sub>16</sub>(<i>n</i>) in terms of <i>x</i>?</p>

<p>1c. Show that lg(2<i>n</i>) = 1 + lg(<i>n</i>) and that
ln(2<i>n</i>) = ln(2) + ln(<i>n
```


top

10. **[3]** Evaluate the following expression which uses reverse-Polish notation.

3 1 3 + 3 + 2 * * 1 + 2 *

Tree Definitions

11. **[3]** For the tree in Figure 11, what are the ancestors of node D and what are the descendants of node B?

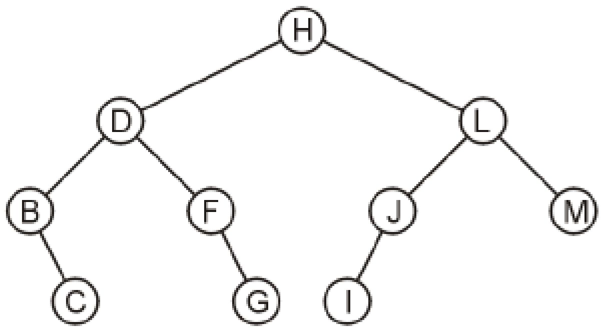

Figure 11. A binary search tree.

12. **[2]** Given a perfect tree with *n* elements, using Landau symbols, what is the maximum number of ancestors a node can have and what is the maximum number of descendants a node can have?

Tree Traversals

13. **[5]** Perform pre- and post-order depth-first traversals and a breadth-first traversal of the general tree in Figure 13. Print the nodes in the order in which they are visited in Tables 13a, 13b, and 13c, respectively.

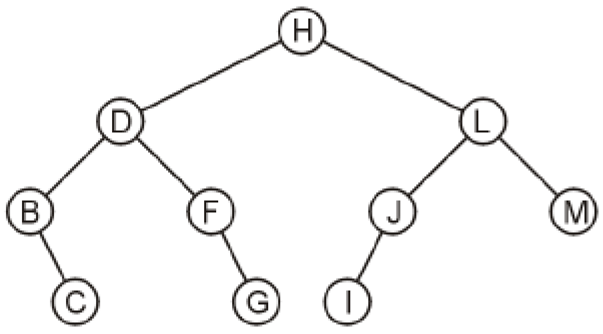

Figure 13. A binary search tree.

Table 13a. Pre-order depth-first traversal.

Table 13b. Post-order depth-first traversal.

Table 13c. Breadth-first traversal.

14. **[3]** Suppose we have a complete binary tree which is also a binary search tree. We can represent the complete tree as an array by performing a breadth-first traversal. Suppose we are doing arbitrary insertions and deletions. Describe, at a minimum, using the appropriate Landau symbol, how much work would be required to insert an element at an arbitrary location. You don't need to prove your argument, an example is sufficient. An example tree is shown in Figure 14.

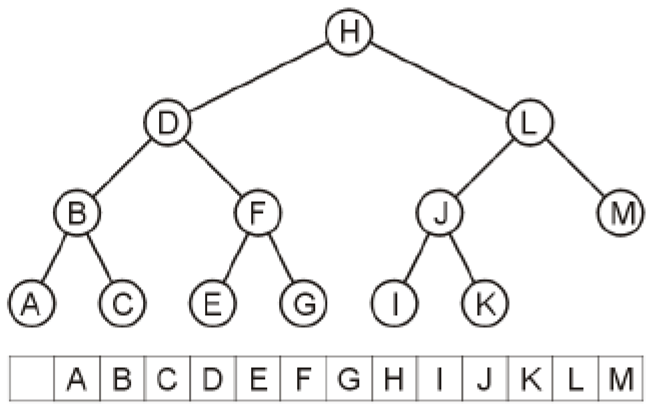

Figure 14. A complete binary search tree.

Binary Search Trees

15. **[4]** Without any balancing, insert the elements 9, 2, 5, 7, 6, 1, 3, 4, 8, 0 into an initially-empty binary search tree. The elements **must** be inserted in the given order.

16. **[4]** When deleting an element from a binary search tree, it can be either a full node, a node with one child, or a leaf node. In the first case, we must copy the appropriate element from the right sub-tree and then delete that element from the right sub-tree. Justify whether or not it is possible that the node deleted from the right sub-tree is also a full node.

17. **[3]** Show the result of deleting the node containing 19 from the binary search tree shown in Figure 17 as described in class.

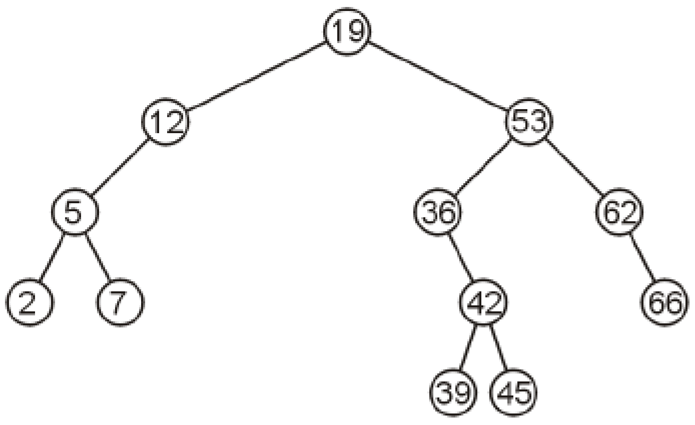

Figure 17. A binary search tree.

Unix

18. **[2]** Show how you would, in Unix, compile the file **`Single_listIntDriver.cpp`** and redirect the contents of the file **`sli`** to determine whether or not it passes its tests.