ECE 250
Data Structures and Algorithms
MIDTERM EXAMINATION

2007-10-25/5:30-7:00

The examination is out of 66 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.

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]** List four of the six relationships which will be viewed in this class.

2. **[2]** In class we showed that Q defines an equivalence relation (where two functions *a* and *b* are related, *i.e.* $a \sim b$, if $a = Q(b)$) but using big-**O** does not. This was because one or more of the three requirements of an equivalence relation

i. $a \sim a$
ii. $a \sim b$ if and only $b \sim a$
iii. $a \sim b$ and $b \sim c$ implies that $a \sim c$

are not satisfied by big-**O**. Which one(s) were not satisfied? Circle your answer(s).

Runtime Analysis

3. **[6]** Place the following six functions into Table 3 in the order $f_1, ..., f_6$ so that $f_k = \mathbf{O}(f_{k+1})$. (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.

$3n + 100$	$1 + 17n^2 \ln(n)$	$5n + 3n^2$
6	$\log_{10}(n)$	$48 + 19n^2 \lg(n)$

Table 3.

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

5. **[2]** Argue that $n^{\lg(3)} = \mathbf{o}(n^2)$.

6. **[5]** Show, using a proof by induction, that $\sum_{i=1}^{n} i2^i = 2 + (n-1)2^{n+1}$. The outline is provided for you.

Show it is true for $n = 1$:

Assume it is true for $n = k$, *i.e.*, assume that $\sum_{i=1}^{k} i2^i = 2 + (k-1)2^{k+1}$ is true.

Show that this implies that it is true when $n = k + 1$.

7. **[6]** A binary search of an ordered list of size n has a run time which is **O**(ln(n)) while a linear search of the same ordered list is **O**(n). What is the run time of the following two hybrids of these two algorithms? Give some justification for your answer.

a. Perform five steps of a binary search and then, if necessary, us a linear search.

b. Perform a binary search until the size of the list is five or less and then perform a linear search.

Arrays and Linked Lists

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

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

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

           SingleList & operator = ( const SingleList & rhs );

           int size() const;
           bool empty() const;
           Object front() const;
           Object back() const;
           SingleNode<Object> * head() const;
           SingleNode<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 SingleNode {
    private:
           Object       element;
           SingleNode * next_node;

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

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

8. **[8]** Implement a member function `push_third` which inserts a new object into the third location of your singly-linked list class. The specifications are:

a. If the linked list has fewer than two entries, it must throw an underflow exception,
b. Otherwise, it must place the argument in a new singly-linked node in the third location of the linked list.

You **may not** use the member functions `push_front` and `pop_front`.

```
template <typename Object>
void SingleList<Object>::push_third( const Object & obj ) {











}
```

9. **[3]** What are the problems with the following implementation of `push_third`?

```
template <typename Object>
void SingleList<Object>::push_third( const Object & obj ) {
     Object tmp1 = pop_front();
     Object tmp2 = pop_front();
     push_front( obj );
     push_front( tmp1 );
     push_front( tmp2 );
}
```

Stacks and Queues

10. **[6]** In class, we showed how a stack may be used to track matching opening and closing delimiters (that is, **(**, **)**, **[**, **]**, **{**, and **}**) while stepping through a piece of C++ code where each time an opening delimiter is passed, it is placed on the stack, and each time a closing delimiter is passed, it is compared to the opening delimiter on the top of the stack.

a. [1] What is done if the closing delimiter and the top of the stack match (for example, the closing delimiter **)** matches **(**)?

b. [1] Give an example of a useful error message you (as the author of a compiler) would print if the closing delimiter did not match the top of the stack?

c. [4] Determine what the state of such a stack is at the end of this piece of code (with line numbers) and place your answer into Table 10.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      const int M = 30;
6      int N = 1024;
7      int parent[N], height[N], depth[N];
8      int maxheights[100];
9
10     for ( int i = 0; i < 100; ++i ) {
11         maxheights[i] = 0;
12     }
13
14     double minmeandepth = 1e300;
15     double meandepths = 0;
16     double maxmeandepth = 0;
17
18     for ( int j = 0; j < M; ++j ) {
19         for ( int i = 0; i < N; ++i ) {
20             parent[i] = -1;
21             height[i] = 0;
22         }
23
24         for ( int i = 0; i < N - 1; ++i ) {
25             while ( true ) {
26                 int p1 = rand() & (N - 1);
27                 int p2 = rand() & (N - 1);
28
28                 int s1 = p1;
30                 int s2 = p2;
31
32                 while ( parent[s1
```

Table 10. The delimiter stack for stepping through C++ code.

Bonus **[2]**: What is the most serious problem with this code??? (Very short answer!!!)

Tree Definitions

11. **[4]** Given any two nodes within a general tree, they will share one or more common ancestors. Of these ancestors, there is one unique ancestor which is deepest in the tree. Given the tree shown in Figure 11, what is the deepest common ancestor of the nodes containing:

a. I and F
b. D and K
c. A and G
d. G and K

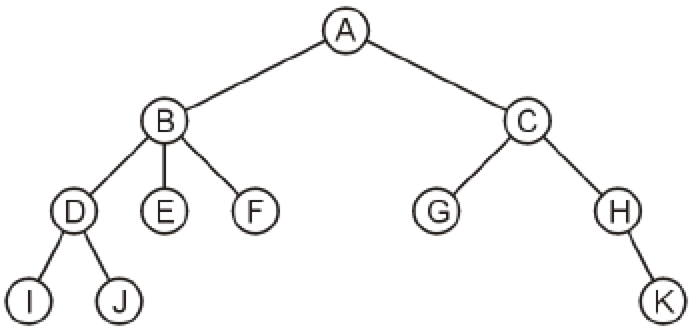

Figure 11. A general tree.

Tree Traversals

12. **[5]** Perform pre- and post-order depth-first traversals and a breadth-first traversal of the general tree in Figure 12. Print the nodes in the order in which they are visited in Tables 12a, 12b, and 12c, respectively.

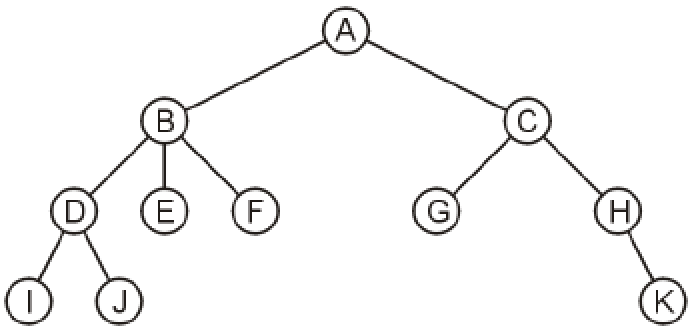

Figure 12. A general tree.

Table 12a. Pre-order depth-first traversal.

Table 12b. Post-order depth-first traversal.

Table 12c. Breadth-first traversal.

Binary Search Trees

13. **[3]** Without any balancing, insert the elements 2, 4, 8, 6, 0, 5, 3, 9, 1, 7 into an initially-empty binary search tree. The elements **must** be inserted in the given order.

14. **[2]** Assuming that the binary search tree (storing integers) shown in Figure 14 does not have any duplicate elements, give the range (in the form *m* – *n*) of all values which could occupy the blank node.

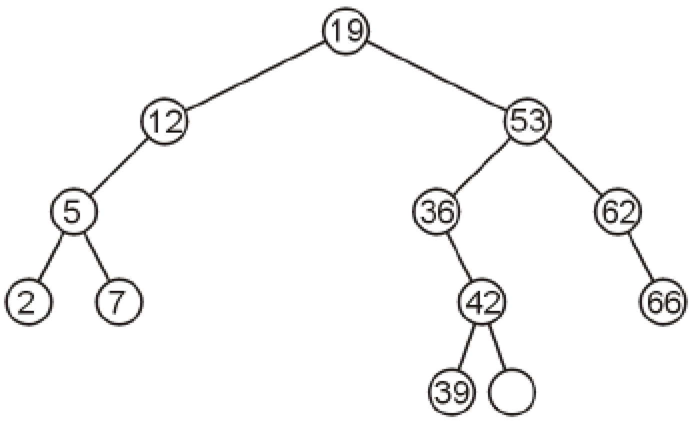

Figure 14. A binary search tree storing non-duplicate integers.

15. **[2]** Show the result of deleting the node containing 12 from the binary search tree shown in Figure 15. You need not redraw the right sub-tree of the root.

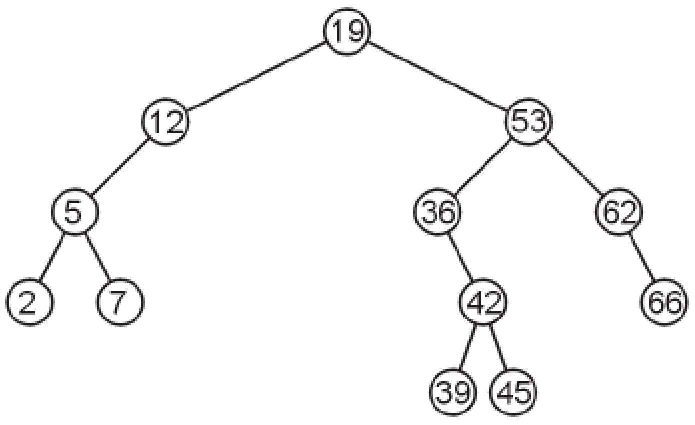

Figure 15. A binary search tree.

16. **[3]** Show the result of deleting the node containing 53 from the binary search tree shown in Figure 16. You should copy the appropriate element from the left sub-tree. You need not redraw the left sub-tree of the root.

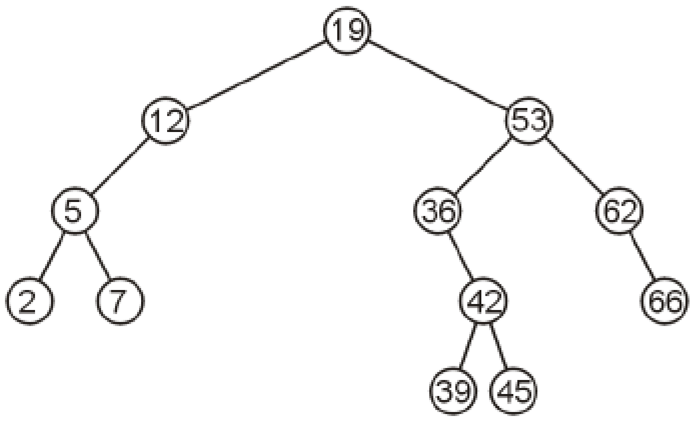

Figure 16. A binary search tree.

Unix

17. **[3]** Write the commands necessary to:
   a. *mak*e a *dir*ectory **`myproject`**,
   b. *c*hange into that *d*irectory,
   c. edit two files **`MyArray.h`** and **`MySupport.h`** (two commands, your choice of editor),
   d. create a *t*ape *ar*chive using **`tar –cvf Array.tar`** of the two files,
   e. zip the files using *g*nu *zip*.