ECE 250
Data Structures and Algorithms
MIDTERM EXAMINATION

2008-10-23/5:15-6:45
REC-200, EVI-350, RCH-106, HH-139

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:

Name: _____

UW Student ID Number: _____

Signature: _____

Relationships

1. **[4]** Given the numbers -1, 0, 1, 2, 3, 4, 5, 6, group these into equivalence classes based on $a \sim b$ if **`(a % 4) == (b % 4)`** where **`%`** is the C++ modulo/remainder operator.

Runtime Analysis

2. **[6]** List six functions in Table 2 in the order $f_0 = 1$, f_1, ..., f_6, $f_7 = n^3$ so that $f_k = \mathbf{o}(f_{k+1})$. Each inversion, error, or missing entry results in -1 with a minimum score of 0.

Table 2.

1							n^3

3. **[2]** Show that $n^{1/2} \ln(n) = \mathbf{o}(n)$ by using limits.

4. **[8]** What is the run time of each of the following loops? The standard function **`min`** retruns the minimum of the two arguments.

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

for ( int i = 0; i < n; ++i ) {
     for ( int j = 0; j < std::min( 10, i ); ++j ) {
        ++sum;
    }
}



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

5. **[4]** A perfect quaternary tree of height h is either a single node if $h = 0$ or a root node with four sub-trees each of which is a perfect quaternary tree of height $h - 1$. Use induction to show that the number of nodes in a perfect quaternary tree is $4^{h+1}/3 - 1/3$.

Note that the formula, while involving fractions, always produces an integer: 1, 5, 21, 85, 341, 1365, ...

6. **[3]** Show by induction that the following sum is $n(n + 1)$.

$$\sum_{k=1}^{n} 2k$$

Arrays and Linked Lists

7. **[8]** A doubly linked list contains nodes which, in addition to a pointer storing the address of the next node, have a pointer which stores the address of the previous node in the list. The first node has a previous pointer which is 0. Implement the functions push front and pop front which preserve the state of the doubly linked list. Functionality is expected to be similar to that of the single list class in Project 1.

```
template <typename Object>
class Double_node {
    private:
        Object elem;             // element
        Double_node *nn;         // next node
        Double_node *pn;         // previous node
    public:
        Double_node(
            const Object & e, Double_node *n = 0, Double_node *p = 0
        ):elem(e), nn(n), pn(p) {
            // empty
        }
        Object retrieve() const;
        Double_node *next() const;
        Double_node *prev() const;
};

template <typename Object>
class Double_list {
    private:
        Double_node<Object> *lh;        // list_head
        Double_node<Object> *lt;        // list_tail
        int count;
    public:
        Double_list():lh(0), lt(0), count(0)
        { /* empty */ }

        // functions similar in name and functionality to Project 1
};

template <typename Object>
void Double_list<Object>::push_front( const Object & obj ) {









}

template <typename Object>
Object Double_list<Object>::pop_front() {









}
```

Stacks, Queues, and Deques

8. **[4]** Why would a singly linked list (for example, your implementation in Project 1) be an inappropriate data structure to implement a deque? Provide quantitative analysis in your arguments.

Tree Definitions

9. **[1]** A node A is a descendant of B if there is a path from ____ to ____ .

10. **[5]** Give the best description of the relationship between the node D and the nodes shown Figure 10 as listed in Table 10. Place your answer in the table.

Table 10. Nodes in Figure 10

Node	**Relationship: D is _____ of *Node***
B	
E	
I	
A	
C	

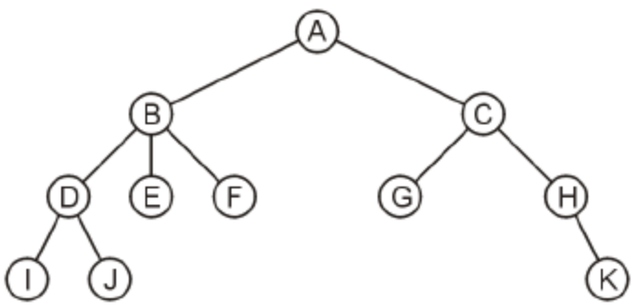

Figure 10. A general tree.

Tree Traversals

11. **[2]** Perform a pre-order depth-first traversal on the binary tree shown in Figure 11 and put your answer in Table 11.

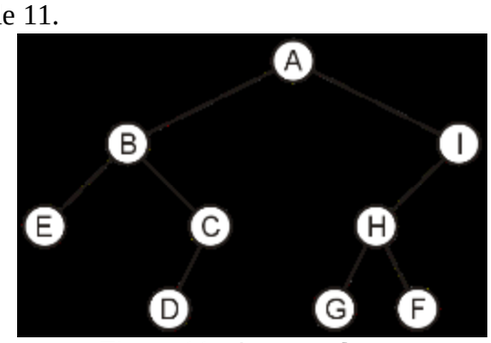

Figure 11. A general tree.

Table 11. Pre-order depth-first traversal.

12. **[2]** Figure 12 shows a tree missing four entries.

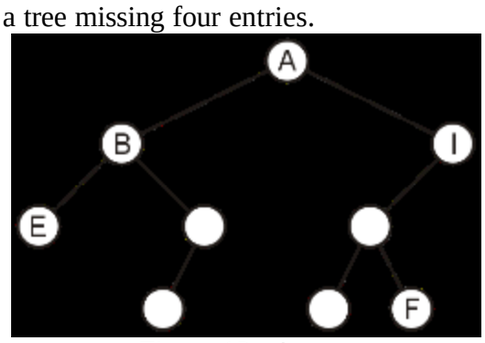

Figure 12. A tree.

Table 12 shows a breadth-first traversal of the tree. Fill in the missing nodes.

Table 12. Breadth-first traversal.

A	B	I	E	K	V	S	V	F

13. **[2]** Figure 13 shows a tree missing four nodes.

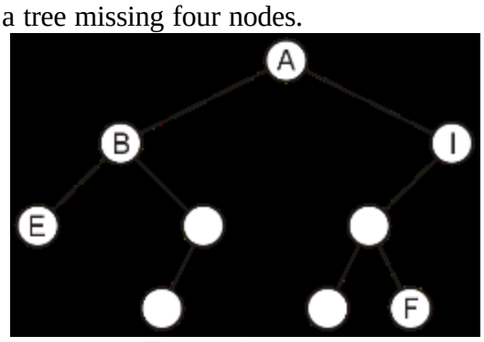

Figure 13. A tree.

Table 13 shows a post-order depth-first traversal of the tree. Fill in the missing nodes.

Table 13. Post-order depth-first traversal.

E	T	R	B	P	F	R	I	A

Binary Search Trees

14. **[4]** Without any balancing, insert the elements 1, 0, 7, 3, 6, 8, 5, 9, 4, 2 into an initially-empty binary search tree. The elements **must** be inserted in the given order.

AVL Trees

15. **[7]** Perform the following insertions into the given AVL trees and perform the appropriate rotations to rebalance the tree.

15a. Add 11 to the AVL tree shown in Figure 15.

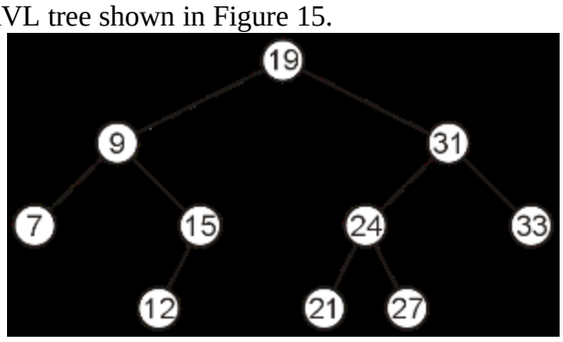

Figure 15a. An AVL tree.

15b. Add 20 to the AVL tree shown in Figure 15b.

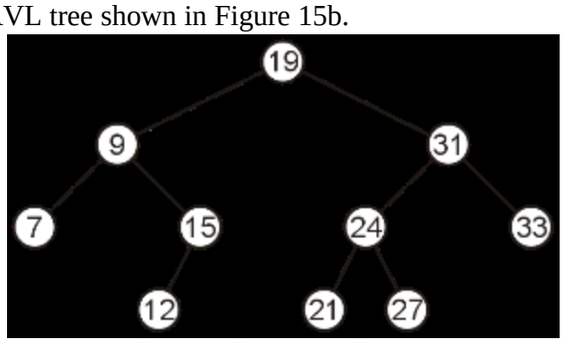

Figure 15b. An AVL tree.

15c. Add 25 to the AVL tree shown in Figure 15c.

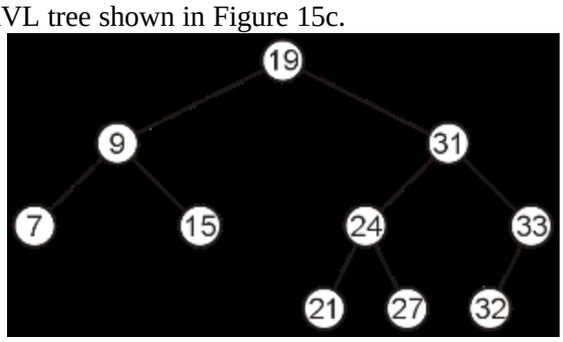

Figure 15c. An AVL tree.

16. **[4]** Remove the node 19 from the AVL tree shown in Figure 16 using the technique shown in class and rebalance the resulting tree.

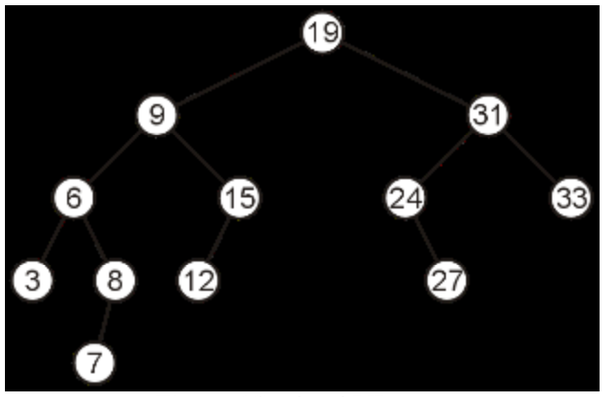

Figure 16. An AVL tree.

Unix

17. **[3]** Write the Unix commands necessary to submit Project 1 assuming you have copied the files relevant header files to Unix. You should end up with an appropriately zipped file.