CS 216 Exam 1 – Spring 2001 - Solution

Part 1 – Closed Book

Name _______________________  Section _______________________

Email Address _________________  Student ID # _________________

Pledge:

This exam is in two parts. Once you are finished with part 1, hand it in, and you will be given part 2. Part 1 is worth 70 points and Part 2 is worth 30 points. You will have an hour and fifty minutes total to complete both parts.

Part 1: Closed note, closed book. You are not to speak with anyone except the Instructor or a teaching assistant for any reason except an emergency during the exam. No calculators are allowed.

Part 2: Open book, open notes. You may not speak with anyone except the Instructor or a teaching assistant for any reason except an emergency during the exam. Calculators are allowed. You may not borrow notes or textbooks from other students.

Obviously, you are not to discuss the exam with anyone until after everyone has taken it! (Tuesday at 5:30pm)

Good Luck!

<table>
<thead>
<tr>
<th>PART</th>
<th>MAX</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
CS 216 Exam 1 – Part I – Closed Book

1. (8 points) Define the terms. Be as specific as possible.

(a) binary search tree

Tree where each node X has these properties:
   a) X has at most two children
   b) all of the nodes in X’s left subtree have value < X
   c) all of the nodes in X’s right subtree have value > X

(b) depth of a node in a tree – the length of the path (number of edges) from the root to that node.

(c) AVL property – each node in an AVL tree must have this property: the height of the left subtree and right subtree differ by at most 1.

(d) mantissa – the fractional part of a floating point number (the other part is the exponent)

2. a) (2 points) How many bits are required to represent integers in the range –256 to 255 in two’s complement notation?

9 bits

b) (2 points) Name two reasons why two’s complement representation is preferable over sign magnitude.

- only one zero
  – arithmetic is easy
3. (9 points) What is the representation of each of the following in the indicated radix? Be sure to show your work.

(a) 36128 in hex

=> 011 110 001 010 written as binary, regroup as hex:

0111 1000 1010 => 0x78A

(b) 1IA19 in decimal

1*19^2 + 18*19 + 10*1 = 713_{10}

(c) 109_{10} in radix 6

109/6 = 18 rem 1 = d0
18/6 = 3 rem 0 = d1
3/6 = 0 rem 3 = d2

301_6

4. (6 points) Consider the positive binary integer 01001101011111012. Express

a. this binary number in octal - (group bits in 3’s from the right)

0 100 110 101 111 101 => 465758

b. this binary number in hexadecimal - (group bits in 4’s from the right)

0100 1101 0111 1101 => 0x4D7D

c. the negative of the number, assuming a two’s complement representation (same number of bits) – flip all bits, then add one:

10110010100000112
5. (12 points total) Describe the running time of the following pseudocode in Big O notation:

a) for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        a = a + b;
    }
}

O(n^2)

b) for (int i = 0; i < 10000; ++i) {
    for (int j = 0; j < n; ++j) {
        a = a + b;
        b = b + c;
        cout << a + b;
    }
}

O(n)

c) for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        a = b + c;
    }
    for (int k = 0; k < n; ++k) {
        cout << a + b;
    }
}

O(n^2)

d) p = 0;
   while (p < n) {
      A[p] = new int;
      p++;
   }

O(n)
6. (4 points) Consider a binary search tree that is built by inserting the sequence: 
   5, 1, 7, 6, 8, 4, 2, 3 
   into an initially empty binary tree.

Draw the tree constructed by this insertion sequence.

```
      5
     / \
    1   7
   / \ / \ 
  4   6 8   
 / \ /   /   
2   3 9   
```

7. (7 points) Given the following tree:

```
    7
   /|
  1 8
 / \
2   9
```

a) Is it an AVL tree? If not, circle the nodes that violate the AVL property. Why or why not (must answer for any credit)?

   Yes. At each node, the heights of the children differ by at most 1.

b) List the order in which the nodes are visited for a postorder traversal:

   2,1,9,8,7
8. (5 points)
a) Describe 2 different methods of implementing the GetSize() function on a linked list class.

1) Keep track of the number of nodes in the list as you insert them.
2) Count the elements only when GetSize is called.

b) For each method, describe a (general) situation where that method would be preferable to the other method.

1) If you want to know the size often, keep track of it as you add items.
2) If you rarely ever want the size of the list, only count when GetSize is called.

9. (3 points) Draw the BST tree after removing the value 6 from the tree. Use the method we described in class.

(The method described in class was replacing the node(6) with the smallest value in the right subtree(7), and recursively calling remove on that node.)

Before: 

```
    6
   / \  
  1   10
 /     /  
2   7   11
```

After: 

```
    7
   /  
  1   10
 /     /  
2   8   11
```

10. (6 points) I am writing a program that will keep track of student grades for cs216. I will use a student record that holds the name, student number, and grades for each student. Give me one reason I might want to use each of the following data structures AND one reason I might not want to use this structure:

Array of records – no wasted space but must know size of class from beginning

Linked List of records – can grow dynamically but wasted space due to pointers

Binary Search Tree of records – can find student records quickly but wasted space due to pointers

11. (6 points) The 216vector class is implemented using arrays, however, there is no limit on the number of elements you can have in a 216vector. The way we accomplish this is by initially allocating an array of some size, and then when we need more space, we re-allocate another array of double the size of the current one. We then copy the elements of the old 216vector into the new 216vector and then delete the old 216vector.

What will be the worst case running time in big-O notation of each of the following operations? Describe how you came up with that value. [Note: Not written on actual exam. Would be good if answer mentioned what is the worst case.]

a) Pushback (add an element at the end of the 216vector) – O(n)

The worst case is if we have reached capacity and must reallocate a new array of double size, O(n). Then we must copy the old array over O(n), and then delete the old array O(n). We get O(n) + O(n) + O(n) = O(n)

b) operator[] to access an element – constant time - O(1)

It takes constant time to access an element in a C++ array, regardless of whether you are reading or writing the location.

c) Insert_before(int loc, element val) – (inserts val before location loc) - O(n)

The worst case is inserting an element before the first element in the list. In this case all n elements will have to be moved down one location. Also if this new element puts us over capacity, then we are back to the worst case for pushback, where we must reallocate a new array of double size, copy over the old array and then delete the old array.
12. (12 points) Given a 16 bit word machine with IEEE floating point format depicted as:

```
+-------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   sign   |   exp |   mantissa |
+-------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|            0 1 . . . 5 6 . . . . . 15 |
```

Assume that the mantissa is 10 bits (11 with hidden) and that the exponent is 5 bits with excess 8 format. What is the representation (in binary and in hex) of the exponent in the representation for

a) $23.0_{10}$

1) Convert 23 to binary: $23/2 = 11$ rem 1 = $d_0$
   $11/2 = 5$ rem 1 = $d_1$
   $5/2 = 2$ rem 1 = $d_2$
   $2/2 = 1$ rem 0 = $d_3$
   $1/2 = 0$ rem 1 = $d_4$ => $10111_2$

2) Normalize this (to get 1 digit to the left of the radix point) => $1.0111 \times 2^4$

3) Convert exponent of 4 into excess 8 notation => $4 + 8 = 12$
   => $1100_2 = 0xC$

b) $3.625 \times 2^{-5}$ (note that the 3.625 is in decimal)

1) Convert 3.625 to binary: We have 3 => $11.0_2$
   Plus we have .625 => $1*\frac{1}{2} + 0*\frac{1}{4} + 1*\frac{1}{8}$
   => $0.101_2$
   So, => $11.101_2 \times 2^{-5}$

2) Normalize this (to get 1 digit to the left of the radix point) => $1.11012 \times 2^{-4}$

3) Convert exponent of -4 into excess 8 notation => $-4 + 8 = 4$

   => $100_2 = 0x4$
13. (18 points) Show how you would implement a queue using two stacks.

a) (4 points) Write the general idea of how your enqueue and dequeue functions would work.

Enqueue – push element onto stack #1

Dequeue – pop all elements off of stack #1 and push onto stack #2
- pop top element off of stack #2 and return it as the dequeued value
- pop the remainder of elements off of stack #2 and push back onto stack #1.

b) (8 points) Given the .h file for the Queue class, and the .h file for a Stack class as described in Weiss on p. 94, write C++ code for the enqueue and dequeue methods. The .h file for the Stack class is reproduced on the next page for your convenience.

Add any extra data members to the Queue class if needed for your implementation

[See next page for answer]

c) (6 points) What is the big-O running time of your enqueue function? Your dequeue function? Explain briefly how you got your answer.

Enqueue = O(1),
Enqueue just calls stack1.push. For the linked list implementation of a stack, push takes constant time.

Dequeue = O(n)

Dequeue calls topAndPop (O(1)) and then push (O(1)) for each element in Stack1. => O(n)
Then it calls topAndPop one more time to get the front item on the queue. => O(1)
Then it calls topAndPop (O(1)) and then push (O(1)) for each element in Stack2. => O(n)
O(n) + O(1) + O(n) = O(n)
// Stack Implemented as a linked list. Reproduced here for your convenience. From p. 94 in Weiss.

template <class Object>
class Stack
{

  public:
    Stack( );
    Stack( const Stack & rhs );
    ~Stack( );

    bool isEmpty( ) const;
    bool isFull( ) const;
    const Object & top( ) const;

    void makeEmpty( );
    void pop( );
    void push( const Object & x );
    Object topAndPop( );

    const Stack & operator=( const Stack & rhs );

  private:
    struct ListNode
    {
      Object   element;
      ListNode *next;

      ListNode( const Object & theElement,
                ListNode * n = NULL )
      : element( theElement ), next( n ) { }
    }

    ListNode *topOfStack;
};
// This is the new class, a queue implemented with two
// stacks: (based on the Queue class interface on p. 113 of
// Weiss) Implement the enqueue and dequeue methods on the next
// page. Add any additional private data members (if any)
// necessary for your implementation.

template <class Object>
  class Queue
  {
    public:
      Queue( int capacity = 10 );

      bool isEmpty( ) const;
      bool isFull( ) const;
      const Object & getFront( ) const;

      void makeEmpty( );
      Object dequeue( );
      void enqueue( const Object & x );

    private:
      Stack<Object> Stack1;
      Stack<Object> Stack2;

      // Add other data members as needed:

      int currentsize; // optional

      void increment( int & x );
  };}
Write your code for the enqueue and dequeue methods on this page. If you need to add any more private data members to the Queue class, add them on the previous page.

```cpp
template <class Object>
void Queue<Object>::enqueue( const Object & x )
{
    if( isFull() )
        throw Overflow();
    Stack1.push(x);
    currentSize++;
}

template <class Object>
Object Queue<Object>::dequeue( )
{
    if( isEmpty( ) )
        throw Underflow( );

    currentSize--;   

    // pop all elements off of stack 1
    // and push them onto stack 2.
    while (!Stack1.isEmpty()) {
        Stack2.push(Stack1.topAndPop());
    }

    // remove front object
    Object frontItem = Stack2.topAndPop();

    // push elements back onto stack 1
    while (!Stack2.isEmtpy()) {
        Stack1.push(Stack2.topAndPop());
    }

    return frontItem;
}