This exam is closed note, closed book. You will have an hour and fifty minutes total to complete the exam. You may NOT use calculators.

It is an Honor Code violation to discuss this exam with ANYONE (including other students who have already taken the exam) until after 5:30pm Tuesday, Feb 24, 2004.

Good Luck!!

Write and sign pledge after taking the exam:
CS 216 Exam 1

**Define** each term and give an **example** that explains it:

1. (2 points) Assembler (the assembler program, not “assembly code”)

   The program that takes assembly language as input and converts it into machine code.
   (Example: Load I → 0011 0000 0000 0100)

2. (2 points) Binary search tree

   A tree where every node has 1) at most two children, 2) every value in the left subtree is less than the value in the node, every value in the right subtree is greater than the value in the node. (Example omitted)

3. (2 points) Enqueue operation

   Operation that appends an element onto a queue data structure. Values are returned with a dequeue operation in a first in first out order.
   Example: MyQueue.enqueue(4), MyQueue.enqueue(5), MyQueueDequeue() will return a 4.

4. (2 points) Leaf node in a tree

   Node that has no children.
   (Example omitted)
5. (2 points) The two rules of “big-Oh math” described in class were: (these were ways to simplify a big-Oh expression)

1) Pick the term that dominates (the higher order term) [Ex. O(N^3 + N^2) \Rightarrow O(N^3)]
2) Ignore constants. [Ex. O(1000 N^3 + 2000) \Rightarrow O(N^3)]

6. (3 points) In class we discussed comparing the running times of two solutions to a problem as a way of deciding which solution to choose. List three problems we came up with to that approach to deciding between two solutions.

Besides the relative inefficiency of having two people go to the trouble of implementing, testing, and benchmarking two different solutions, we mentioned that factors such as: data set used, hardware, OS, compiler, programming language, programmer skill, and compiler optimizations may have an affect on the running time of the two solutions. We would like any solution we choose to be independent of these factors. Any solution we choose should be able to take advantage of improvements in for example hardware or OS speed.

7. (6 points) The Virginia Board of elections has asked you to create a data structure that will keep track of voters in Virginia. Each record should consist of the voter’s last name and their county of residence.

A colleague suggests that you use a fixed size array of C++ characters to represent the voter’s name. You point out that using a fixed size array will not be large enough in all cases, so you propose that a singly linked list of C++ characters be used. In both cases an extra null character at the end of the name is NOT required.

a) How many bytes total will be required to represent a name that is 9 letters long in your scheme? Be sure to include a pointer to the head of the list in your calculations. Assume that a dummy node is not used. Show your work!

4 bytes (the head pointer) + 9 * (1 byte (the char) + 4 bytes (next ptr)) = 49 bytes

If all values are aligned on 4 byte boundaries:
4 bytes (the head pointer) + 9 * (4 bytes (the char) + 4 bytes (next ptr)) = 76 bytes

b) Your boss throws out both suggestions and demands that you implement voter last name as a pointer to a heap allocated array of characters that will be exactly the size of the voter’s last name (again, no null character is needed at the end). How many bytes total will this scheme require for a name that is 7 letters long? Show your work!

4 bytes (the head pointer) + 7 * (1 byte (the char)) = 11 bytes
c) If there are 134 different counties in Virginia, and we assign each county a number from 1 to 134, how many bits would be required to represent a voter’s county of residence (assuming we could allocate exactly the number of bits needed)?

\[128 \text{ < } 134 \text{ values } \text{ < } 256\]

\[2^7 \text{ } < \text{ } 134 \text{ values } < 2^8\]

So we need 8 bits.

8. (4 points) Given the following infix expression: \(((k + j) \times (a - g)) / (e + b)\)

a) Write it as a prefix expression:

```
/ * + k j a g e b
```

b) Write it as a postfix expression:

```
k j a g * e b + /
```

[Diagram of the expression tree]
9. (10 points total) Describe the running time of the following pseudocode in Big-Oh notation in terms of the variable \( n \). Assume all variables used have been declared.

*Show your work for partial credit.*

```c
int foo(int k) {
    int sum;
    if (k > 1) {
        for (int i = 0; i < 500; ++i) {
            sum = sum + (i * i);
        }
    }
    return sum;
}

int bar(int k) {
    if (k < 100)
        return 0;
    else {
        for (int i = 0; i < k; ++i) {
            sum += i;
        }
        return sum;
    }
}

a) answ = foo(n); \quad O(1)

b) answ = bar(n); \quad O(n)

c) for (int j = 0; j < n; ++j) {
    cin >> val;
    for (int i = 0; i < j; ++i) {
        b = b * val;
        for (int k = 0; k < 50; ++k)
            c = b + c;
    }
}

d) for (int i = 0; i < n * n; ++i) {
    if (i < n) {
        for (int j = 0; j < i; ++j)
            j >> cout;
    }
}

e) for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i * i; ++j) {
        sum = sum + i;
    }
    for (int k = 0; k < n + 1000; ++k) {
        a[k] = a[k] + sum;
    }
}
```
10. (6 points total) What is the representation of each of the following in the indicated radix? Be sure to show your work.

a) $23_9$ in decimal

$$= 2 \times 9^1 + 3 \times 9^0$$
$$= 18 + 3 = 21$$

b) $2317_8$ in hex

$$= 2 \times 8^3 + 3 \times 8^2 + 1 \times 8^1 + 7 \times 8^0$$
$$= 3 \times 4096 + 3 \times 64 + 1 \times 8 + 7 \times 1 = 12288 + 192 + 8 + 7 = 12495_{10}$$

$$12495/16 = ? \text{ rem } ?$$

etc. \hspace{0.5cm} = 4CF_{16}

$$010 \ 011 \ 001 \ 111 \rightarrow 0100 \ 1100 \ 1111 \ (\text{group the digits to get:} \ 4CF_{16})$$

c) $2A_{11}$ in radix 10

$$= 2 \times 11^1 + 10 \times 11^0$$
$$= 2 \times 11 + 10 \times 1 = 22 + 10 = 32_{10}$$

11. (6 points total) Consider the positive binary integer represented in two’s complement: $0111110110001010_2$.

a) Express this binary number in octal

$$0111 \ 110 \ 110 \ 001 \ 010_2 = \ 76612_8$$

b. Express this binary number in hexadecimal

$$0111 \ 1101 \ 1000 \ 1010_2 = \ 0x \ 7D8A$$

c. Negate the number (i.e. give the two’s complement representation of a negative version of the same number) Use the same number of bits.

$$1000001001110110_2$$
12. (3 points) Draw the binary search tree created by inserting these values in this order:

```
3 4 0 2 8 6 5 1 7 9
```

13. (2 points) Give a pre-order traversal of your tree shown above:

```
3 0 2 1 4 8 6 5 7 9
```

14. (2 points) Give a post-order traversal of your tree shown above:

```
1 2 0 5 7 6 9 8 4 3
```

15. (3 points) Delete the root of the tree shown above using one of the two methods described in class. Draw the new tree here:

   Replace with smallest on right

```
4

0

2

1

0 2 6 9

8

5 7

1 5 6 7
```
Or replace with largest on left

```
0 -> 1 -> 2
4 -> 5 -> 6
7 -> 8 -> 9
```
16. For each operation below give: 1) How you would most efficiently implement the operation, 2) Describe the worst case scenario (e.g. “The worst case occurs when the value you are looking for is not in the list”) and 3) What is the worst case Big-Oh running time of this scenario. State any assumptions you make.

a) (3 points) Print the 5 largest values in a list stored as an unsorted linked list.

1) Search the list 5 times(each time remember that you printed out) or keep variables that record the top 5, and modify them as you examine the entire list just once.
2) There is no worst case, you must always examine the entire list. However, if list was arranged in ascending order, then a lot of modifies to the 5 variables would have to happen.
3) O(N)

b) (3 points) Find the maximum value stored in a BST (non-AVL).

1) Keep following right pointers, until there are no more to follow.
2) Values inserted in increasing order
3) O(N)

c) (3 points) Enqueue a value into a queue implemented as an array.

1) a. increment tail index (For circularly, wrap around if needed, don’t let tail cross over head)
   b. write in new value
2) Array is full and need to re-alloc a new array and copy over old values.
   If you aren’t going to do re-alloc, then all cases are pretty much the same.
3) All cases O(1). Re-alloc: O(N)
17. (3 points) Given the following tree:

```
  40
 /   \
22    77
 /     /  \
17    30    96
 /  \
12   
```

Is it an AVL tree? If not, circle the node(s) where the AVL property is violated. Why or why not (must answer for any credit)?

Yes, at each node the AVL property holds (it is a BST tree and the heights of the subtrees differ by at most 1).

18. (3 points) Given the following AVL tree:

```
  8
 /   \
5     9
 /     /  \
3     16
     /  \
    4   
```

Insert the value 4 into the AVL tree above, doing any necessary rotations to maintain the AVL property.

```
  8
 /   \
5     9
 /     /  \
3     16
     /  \
    4   
```

first rotation

```
  8
 /   \
5     9
 /     /  \
3     4
     /  \
    16
```

second rotation

```
  8
 /   \
5     9
 /     /  \
3     4
     /  \
    16
```

Exam 1 CS216 – Anderson – Spring 2004 Page 10 of 15
(3 points) Given the following AVL tree:

Insert the value 9 into the AVL tree above, doing any necessary rotations to maintain the AVL property.

No rotations needed (see above).
20. (7 points) Assume we are using the 32-bit IEEE single precision floating point format as described in class and used in lab. The mantissa has 24 bits including the hidden bit. There is one sign bit and there are eight exponent bits. The exponent is stored in excess 127.

What decimal floating point number is represented by the following 32 bits? SHOW YOUR WORK!

1001 0110 1011 0000 0000 0000 0000 0000

a) Is this a positive or negative number?
   Negative. (sign bit is 1)

b) What is the exponent (in base 10)?
   \[00101101_2 = 45_{10}\]
   \[45-127 = -82\]

c) What is the value of the mantissa (in base 10)
   \[1.011_2\]
   \[1 + 1/4 + 1/8 = 1 + 0.25 + 0.125\]
   \[=1.375\]

d) What is the total value?
   Note: you may leave your answer in the form: \[value_{10} \times base^{exponent}\]
   Where you specify value, base and exponent.
   \[-1.375 \times 2^{-82}\]
21. (24 points) For this question you are to implement a two-ended stack of ints. We will define a two-ended stack as a list in which entries can be pushed or popped from either the first or last position of the list, but no changes may be made elsewhere in the list. We will define the fundamental operations on a two-ended stack to be:

push_front – pushes an item on the front of the two-ended stack
push_back - pushes an item on the back of the two-ended stack
pop_front – removes and returns an item from the front of the two-ended stack
pop_back – removes and returns an item from the back of the two-ended stack

Your assignment is to implement the TEStack (two-ended stack) methods listed below using a singly-linked list. You should use the header file provided below. You do not need to implement copy constructors, destructors, or operator= for these classes, but otherwise the routines you implement should handle memory management appropriately.

```cpp
class Node{
    public:
        Node(int val) : Val(val), Next(NULL){};
        int Val;
        Node * Next;
};
class TEStack {
    public:
        TEStack();
        void push_front(int val);
        void push_back(int val);
        int pop_front();  // returns -1 if empty
        int pop_back();  // returns -1 if empty
    private:
        Node *Front;
        Node *Back;
};
```

On the remaining pages implement your answers to parts a) through d). You may answer e) and f) on this page if you wish.

a) (2 points) Implement the constructor for TEStack.
b) (6 points) Implement your push_back method.
c) (6 points) Implement your pop_back method.
d) (6 points) Implement your pop_front method:
e) (2 points) What is the worst case big-O running time of your pop_back method and why?

```
O(n) – you must find the node previous to the back node in order to update the back pointer. This requires iterating through the list.
```
f) (2 points) What is the worst case big-O running time of your pop_front method and why?

```
O(1) – No traversal through the list is needed, just changing of pointers. All operations involved take constant time.
```
TEStack::TEStack(){
    Front = Back = 0;
}

void TEStack::push_back(int val) {
    Node *temp = new Node(val);

    if (!Front) { // Add the first item to the list.
        Front = Back = temp;
    } else {
        Back->Next = temp;
        Back = temp;
    }
}

// This routine was not asked for.
void TEStack::push_front(int val){
    Node *temp = new Node(val);

    if (!Front) { // Add the first item to the list.
        Front = Back = temp;
    } else {
        temp->Next = Front;
        Front = temp;
    }
}

int TEStack::pop_back(){
    Node *temp;
    int return_val;
    if (Back) { // The list is not null
        if (Front==Back) { // There is only one item in list
            temp = Back;
            Front = Back = 0;
            return_val = temp->Val;
            delete temp;
        } else { // > 1 item in list
            Node *temp_prev;
            for (temp = Front->Next, temp_prev = Front;
                temp != Back;
                temp = temp->Next, temp_prev = temp_prev->Next) {};
            Back = temp_prev;
            Back->Next = NULL;
            return_val = temp->Val;
            delete temp;
        }
    } else { // empty list
        return_val = -1;
    }
    return return_val;
}
int TES::pop_front() {
    Node *temp;
    int return_val;
    if (Front) { // The list is not null
        if (Front==Back) { // There is only one item in list
            temp = Front;
            Front = Back = 0;
            return_val = temp->Val;
            delete temp;
        } else { // > 1 item in list
            temp = Front;
            Front = Front->Next;
            return_val = temp->Val;
            delete temp;
        }
    } else { // empty list
        return_val = -1;
    }
    return return_val;
}