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 it) until after 5:30pm Tuesday, Feb 25.

Good Luck!

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

1. (4 points) Complete the definition of big-Oh notation:

Definition: T(N) = O(f(N)) if _____________________________
_____________________________________________________

2. (4 points) Explain the definition above to a programmer who has never heard of it. Feel free to use diagrams.

3. (2 points) How does big-omega (Ω) differ from big-Oh?

4. (3 points) With respect to big-Oh notation, order the following rates of growth from fastest rate of growth to slowest rate of growth: log n, 1000, n log n, n^3, 2^n, n
5. (4 points) Your friend proposes a data structure to represent the temperature in Fahrenheit. She wants to use 5 bits to do this. a) How many different temperatures can we represent? and b) What is the range of positive and negative integer temperatures we can represent in two’s complement?

6. (4 points) In reality, temperatures we need to represent range from -10 F to 90 F. From cs216 we learned that biased notation can represent an asymmetrical range like this. a) What is the **minimum** number of bits we would need to represent this range, and b) What would the bias be, assuming that the temperature will NEVER go below -10 F, but might occasionally go above 90 F?

7. (4 points) Write the following infix expression as a prefix expression:

   $$(((a-g) + (e / b)) * c)$$
8. (10 points total) Describe the running time of the following pseudocode in Big-Oh notation in terms of the variable \( n \). Show your work for partial credit.

a) int test(int k) {
    if (k == 0)
        return 1;
    else
        return (test(k-1) + test(k-1))
}
answ = test(n);

b) int super(int k) {
    for (int i = 0; i < k; ++i) {
        return k;
    }
}
answ = super(n);

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

d) for (int i = 0; i < 217; ++i) {
    if (i < 100) {
        for (int j = 0; j < n; ++j)
            j >> cout;
    }
}

e) b = 1000;
for (int i = 0; i < n * n; ++i) {
    for (int j = 0; j < i; ++j) {
        sum = sum + i;
        for (int k = 0; k < b; ++k)
            a[k] = a[k] + sum;
    }
}
9. (6 points total) What is the representation of each of the following in the indicated radix? Be sure to show your work.

a) $169_{12}$ in decimal

b) $3230_4$ in hex

c) $2A6_{17}$ in radix 10

10. (6 points total) Consider the positive binary integer represented in two’s complement: $0101110101110011_2$.

a) Express this binary number in octal

b. Express this binary number in hexadecimal

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.
11. (3 points) Draw the binary search tree created by inserting these values in this order:

6 8 2 4 3 0 1 9 5 7

12. (2 points) Give a pre-order traversal of your tree shown above:

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

14. (3 points) Delete the root of the tree shown above using one of the two methods described in class. Draw the new tree here:
15. 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) Finding the maximum element in the AVL tree:

b) (3 points) Print all the values in a binary search tree (BST) from smallest to largest.

c) (3 points) Find the minimum value in a stack.
16. (3 points) Given the following tree:

```
  32
 /   \
24     44
|      |
7      29
|      |
26     60
|      |
  31   65
   |
  6
```

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)?

17. (3 points) Given the following AVL tree:

```
  7
 /   \
6     8
|     |
5     9
```

Insert the value 4 into the AVL tree above, doing any necessary rotations to maintain the AVL property.
18. (3 points) Given the following AVL tree:

```
    5
   / \
  3   8
 /   /   |
2   6   7
```

Insert the value 7 into the AVL tree above, doing any necessary rotations to maintain the AVL property.
19. (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!

0011 0011 0011 0000 0000 0000 0000 0000

a) Is this a positive or negative number?

b) What is the exponent (in base 10)?

c) What is the value of the mantissa (in base 10)

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.
20. (21 points total) a) (12 points) This question tests your understanding of queues and pointer manipulation. You must implement a queue ADT in C++. The underlying representation of the queue should be the Node class as described below. Your queue should store integers and should handle errors (printing an error message is fine).

You will be graded mostly on the correctness of the ideas of your solution rather than exact C++ syntax, but your solution should be clear. Correct C++ code is the best way to ensure we understand your solution. You may NOT use the STL in any way for this question.

You should implement all the functions with \( \rightarrow \) in front of them.

```c++
class Node {
public:
    Node(int value) {val=value; prev = next = NULL;};
    int val;
    Node *prev;
    Node *next;
};

class Queue {
public:
    Queue(); // constructor
    void enqueue(int value); // enqueues value into the queue
    int dequeue(); // returns and removes the value at the head of the queue
    bool isEmpty(); // returns true if the queue contains no elements.
private:
    Node *head;
    Node *tail;
    int size;
};
```
[extra space for use in part a) or b)]
b) (5 points) Implement a member function `void printQueue(bool chron)` in C++. `printQueue` should print out all elements currently in the queue in the direction specified by its parameter. If `chron` is true, then print the items out in the order they were originally inserted into the queue. If the `chron` is false, then print the items out in reverse chronological order. For example:

```cpp
Q.enqueue(1);
Q.enqueue(2);
Q.enqueue(3);
Q.printQueue(true); // Should print: 1 2 3
Q.printQueue(false); // Should print: 3 2 1
```

Note: When you return from `printQueue` the original queue should be unchanged!
c) (2 points) Give the worst case big-O running time of your implementation for each of the following operations (in terms of n where n is the number of elements currently in the queue). You MUST explain briefly how you got your answer.

```cpp
enqueue(int value)
```

```cpp
int dequeue()
```

d) (2 points) Give the worst case big-O running time of your implementation of `printQueue` (in terms of n where n is the number of elements currently in the queue). For full credit you **must** explain briefly how you got your answer.