CSCI 211 **NAME** ________________________________

October 2, 2013 **Minutes used:** ______

First Midterm Exam

- This test contains 9 questions worth a total of 82 points.
- Questions 1-5 have short answers and are 6 points each.
- Question 6 is worth 8 points.
- Questions 7-9 require you to write code. They are worth 12, 16, and 16 points.
- You have 50 minutes to complete this exam.
- You may **not** use your text, notes, or any other reference material.
- The test with answers will be posted on the class webpage after the test.
- No electronic devices (music, phone, calculator, etc).
- **Do not turn this page until instructed to do so.**

Test Taking Advice

- The amount of space after a question does not always indicate how long the answer should be. Sometimes I add space so questions fit well on pages.

- Some questions have multiple parts such as “Explain your answer.” Make sure you answer all the parts of each question.

- If you can't answer a question, move on and come back to it later. I often hear something like this, “I spent 40 minutes working on this 10 point problem and left 30 points worth of problems blank.”

- If you have time left over, use it to review your answers. Students who turn tests in early often make trivial mistakes that they would catch if they went back over their answers. Some of my questions are difficult, go back and make sure you understood the question.

- If you don't understand a question ask me during the test. It is too late to ask for clarification after the exam.

1. (6 points) The -> operator is a shortcut for two other operators. For the 2nd statement below, show an equivalent statement using different operators (that is, do the same thing as this line w/o using ->).

```
Bill *my_bill = new Bill();
my_bill->print();
```

```
(*my_bill).print();
```

2. (6 points) The following code compiles and runs without error. What can you deduce about a, b, and c? For example, if *a* must be an integer, write *"a is an integer."*

```
a[b]->c(10.2);
```

a is a list of pointers to objects (or structs)

b is an integer

c is a member function of the objects a is a list of pointers to

c has a single argument that is of type double or float

3. (6 points) You turn in assignment p3 on turnin.ecst.csuchico.edu and find that you fail test t13 because the standard output is incorrect. How can you run the test on a computer other than turnin and figure out what is incorrect about the output from your program? The first two steps are given.

1) Download the test files for p3 and the run_tests script (~tyson/211/tests/p3/tests.tar) to the directory that contains your program.

2) untar the files: $ tar -xvf tests.tar

3) Execute all the tests: $ run_tests videos
-or-
Execute test t13: $ videos < tests/t13.in > results/t13.myout 2>results/t13.myerr

The correct output for test t13 is in the file tests/t13.out
The output your program generated is in the file results/t13.myout

4) In order to determine the difference you must compare these two files. You could use diff:

```
$ diff tests/t13.out results/t13.myout
```

or you could use vimdiff (which will highlight the differences in red):

```
$ vimdiff tests/t13.out results/t13.myout
```

4. (6 points) In the following diagram, each box represents a file with the filename at the top of the box. Add to the code in the boxes the necessary include directives (#include) so the program will compile. Ignore the "..." they are placeholders for real code. Note: class Node is NOT nested in class list.

list.h
```
#include "node.h"



class List
{
   public:
        List();
        void insert(int value);
   private:
        Node *m_head;
};
```

node.h
```




class Node
{
   public:
        Node(int v, Node *next);
        int m_value;
        Node *m_next;

};
```

list.cpp
```
#include "list.h"



List::List()
{
        ...
}

void List::insert(int value)
{
        ...
}
```

node.cpp
```
#include "node.h"



Node::Node(int v, Node *next)
{
        ...
}
```

main.cpp
```
#include "list.h"



int main()
{
        List the_list;

        the_list.insert(42);
        ...
}
```

5. (6 points) Given the following class definition:

```
class Dog
{
    public:
        Dog(int age, int weight)
        {m_age = age; m_weight = weight;}
        void print()
        {cout << "dog's age = " << m_age << " weight = " << m_weight <<endl;}
    private:
        int m_age;
        int m_weight;
};
```

Write code to instantiate a Dog object without using new and then call it's print function. Now do the same thing using new.

```
Dog sam(1,21);
sam.print();

Dog *spot = new Dog(11,103);
spot->print();
```

6. (8 points) What does the following program print? This question is very tricky, be careful!
   Explain your answers or you will not get any points.

```
#include <iostream>
using namespace std;

void f(int value)
{
    value += 1;
}
void g(int &value)
{
    value += 10;
}
void h(int *value)
{
    value += 100;
}

int main()
{
    int i = 17;

    f(i);                          i is passed by value and thus not changed by f()
    cout << i << endl;             prints: 17

    g(i);                          i is passed by reference and thus change by g()
    cout << i << endl;             prints: 27

    h(&i);                         i is passed by address to h(), BUT the function
    cout << i << endl;             changes the local variable value and not the
                                   integer it points to
                                   prints: 27

}
```

For the following questions, use the code on the last page. Tear off last page for easy reference.

7. (12 points) Write the List member function `count_matches()`. This function counts the values in the list that are in the given array. For example, assume the array *values* contains the following numbers [42,86,99] and the list contains the values {86,17, 2, 42, 100, 3, 1, 86}. `count_matches()` would return three because 42 and 86 (which are in the array) appear in the list three times (42 is in the list once, and 86 is in the list twice). Assume the list is NOT sorted and may contain duplicates. The function must not change the list. Size is the number of elements in the `values[]` array.

```
int List::count_matches(int values[], int size)
{
    int count = 0;
    Node *ptr = m_head;
    while (ptr != NULL)
    {
        for (int i = 0; i < size; i++)
        {
            if (values[i] == ptr->m_value)
            {
                count += 1;
            }
        }
        ptr = ptr->m_next;
    }

    return count;
}
```

8. (16 points) Write the function `void List::insert_sorted(int value)` that inserts the given value into the list in such a way that the list is ordered from smallest to largest. Insert the number into the list even if it is already in the list. Assume this list **is** sorted when the function is called.

```
void List::insert_sorted(int value)
{
    // if list is empty or new element belongs at front of list
    if (m_head == NULL || value < m_head->m_value)
    {
       m_head = new Node(value, m_head);
    }
    else
    {
        Node *ptr = m_head;
        // search for the node the new number should be inserted after
        while (ptr->m_next != NULL && ptr->m_next->m_value < value)
        {
           ptr = ptr->m_next;
        }
        assert(ptr != NULL);
        ptr->m_next = new Node(value, ptr->m_next);
     }
}
```

9. (16 points) Write the List member function `remove_values_in_range(int min, int max)`that removes all elements from the list that are in the range min..max (that is, all values greater than or equal to min AND less than or equal to max). For example, if the list contains the values {17,42,86,50} and `remove_values_in_range(40, 60)` is called, after the function call the list will contain the values {17, 86}. Return the number of values removed. Your solution cannot have any memory leaks and must not reorder the elements that are not removed. Assume the list is NOT sorted.

```
int List::remove_values_in_range(int min, int max)
{
  int num_removed = 0;

  // special case: remove values at the head of the list
  while (m_head != NULL && (m_head->m_value >= min && m_head->m_value <= max))
  {
    Node *tmp = m_head;
    m_head = m_head->m_next;
    delete tmp;
    num_removed++;
  }

  // iterate through the nodes, removing values within given range
  Node *ptr = m_head;
  while (ptr->m_next != NULL)
  {
    if (ptr->m_next->m_value >= min && ptr->m_next->m_value <= max)
    {
      Node *tmp = ptr->m_next;
      ptr->m_next = ptr->m_next->m_next;
      delete tmp;
      num_removed++;
    }
    else
    {
      ptr = ptr->m_next;
    }
  }
  return num_removed;
}
```

Use the following class definition for questions 7, 8, and 9. You **may not** alter or add to this class definition.

You may tear this page off so it is easier to reference.

```
class List
{
    public:
       List() {m_head = NULL;}
       ~List();
       void insert_at_front(int value);

       // Functions you have to write
       int count_matches(int values[], int size);
       void insert_sorted(int value);
       int remove_values_in_range(int min, int max);

    private:
       class Node
       {
           public:
              Node (int value, Node *next) {m_value = value; m_next = next;}
              int m_value;
              Node *m_next;
       };
       Node *m_head;
};
```