CSCI 112 NAME ________________________________

March 6, 2009 **Minutes used: ______**

First Midterm Exam

- This test contains 9 questions worth a total of 82 points.
- Questions 1-3 have short answers and are 6 points each.
- Question 4-5 are a bit longer and are worth 8 points.
- Questions 7-10 require you to write code. They are worth 10, 10, 12, and 16 points respectively.
- 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) What is the difference between a class and an object?

   A class is a template for an object, it defines the member functions and the member variables. A class has no memory associated with it.

   An object is an instantiation of a class like in “int i” i is an instantiation of an integer. There is memory associated with an object. There can be many objects of the same class.

2. (6 points) The following program has a single compilation error on the second to the last line. What is the error? How could this code be changed so that error does not occur?

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

class Point
{
    public:
        Point(int x, int y) {m_x = x; m_y = y;}
    private:
        void print() {cout << m_x << "," << m_y << endl;}
        int m_x;
        int m_y;
};

int main()
{
    Point p(86,99);
    p.print();  // the error is issued for this line
}
```

   The member function print() is private. The error message is “Point::print() is private.” It can be fixed by moving the print() in the class definition to before the private:

3. (6 points) Given the following class definition:

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

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

```
Cat mittens(3,16);
mittens.print();

Cat *fluffy = new Cat(11,12);
fluffy->print();
```

4. (8 points) In the following diagram, each box represents a file from the second programming assignment (the filename is at the top of the box). What parts of the Event object go in the event.h and event.cpp files (you can write pseudo code OR an English description)? Write your answer in the boxes.

event.h

event.cpp

main.cpp

```
#include "event.h"

int main()
{
        Event *calendar[MAX];
        ...
            calendar[i] = new Event(...);
        ...
            calendar[i]->print();
        ...
}
```

5. (8 points) What does the following program print? This question is very tricky, be careful! **You must explain your answers for any credit.**

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

int f(int &value)
{
    return value * 2;
}
void g(int &value)
{
    value = value * 2;
}
void h(int *value)
{
    value = value * 2;
}

int main()
{
    int i = 42;

    f(i);
    cout << i << endl;          42, f()'s return value is ignored.  Its argument
                                is passed by reference, but that is irrelevant.
    g(i);
    cout << i << endl;          84, g()'s parameter is a reference parameter
                                so when value is change in g(), i is changed.
    h(&i);
    cout << i << endl;          84, h() modifies the local pointer value, but
                                not the value it points to, thus i does not
                                change.
}
```

6. (10 points) Write the function void print_square(int size) that prints a "square" of numbers with a height and width equal to size and filled with numbers 1..size^2. For example, the following squares would be printed out for square(1), square(2), square(3), and square(4) respectively. You must print a single space between each number and a newline at the end of each line.

```
1
                        void print_square(int size)
1 2                     {
3 4                         int num = 1;
                            for (int row = 0; row < size; row++)
1 2 3                       {
4 5 6                             for (int col = 0; col < size; col++)
7 8 9                             {
                                      cout << num++ << " ";
1 2 3 4                           }
5 6 7 8                           cout << endl;
9 10 11 12                  }
13 14 15 16                 cout << endl;
                        }
```

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

7. (10 points) Write the function List::size() that returns the number of elements in the list. Return 0 if the list is empty.

```
int List::size()
{
    int count = 0;
    Node *ptr = m_head;
    // if the list is empty 0 is returned
    while (ptr != NULL)
    {
        count++; // each time ptr points to non-NULL it is a entry in list
        ptr = ptr->m_next;  // update ptr to point to next entry in list
    }
    return count;
}
```

8. (12 points) Write the function contains_duplicates() that returns true if the List contains any duplicate entires and false if it contains no duplicate entries. Assume the list is sorted from smallest to largest.

```
bool List::contains_duplicates()
{
    // a list with zero or one element cannot have duplicates
    if (m_head == NULL || m_head->m_next == NULL)
    {
        return false;
    }

    // we know there are 2 or more elements
    Node *ptr = m_head;
    while (ptr->m_next != NULL)
    {
        // all duplicates must be neighbors
        if (ptr->m_value == ptr->m_next->m_value)
            return true;
        ptr = ptr->m_next;
    }
    return false;
}
```

9. (16 points) Write a function that removes the specified element from the list. Return true if the target number is found. Return false if the target number is not in the list. Assume the list is not sorted.

```
bool List::delete_value(int target)
{
    // special case, empty list
    if (m_head == NULL)
        return false;

    // special case, the value we are looking for is at head of list
    // this is special because the value of m_head would change
    if (m_head->m_value == target)
    {
        Node *tmp = m_head;
        m_head = m_head->m_next;
        delete tmp;
        return true;
    }

    // the list is not empty, the value we are looking for is not the first
    // find the node BEFORE the one we want to delete
    Node *ptr = m_head;
    while (ptr->m_next != NULL && ptr->m_next->m_value != target)
    {
        ptr = ptr->m_next;
    }

    // did we walk off the end of the list in the above loop
    if (ptr->m_next == NULL)
    {
        // did not find target in the list, cannot remove it
        return false;
    }

    assert(ptr->m_next != NULL);

    Node *tmp = ptr->m_next;
    ptr->m_next = ptr->m_next->m_next;
    delete tmp;
    return true;
}
```

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

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

```
class List
{
   public:
     List() {m_head = NULL;}

     int size();
     bool contains_duplicates();
     bool delete_value(int target);

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