

Name _____ ID _____

1. (4 pts) Inspect the following code fragments for memory leaks:

a)

```
for (int i = 0; i < 10; i++) {  
    ToddClass todd("Test Message", aClass(10,10));  
}
```

Is there a memory leak? YES _____ NO _____

Explain why or why not:

b)

```
ToddClass *t = NULL;  
for (int i = 0; i < 10; i++) {  
    t = new ToddClass("Test thingy");  
}  
delete t;  
t = NULL;
```

Is there a memory leak? YES _____ NO _____

Explain why or why not:

2. (15 pts) Circle the letters of only the statements that are **True**: (Think carefully!)

- a) Abstraction is the process of generalizing a design idea or concept that is intended to be implemented in software.
- b) A class is a software representation of an abstraction.
- c) Abstractions can only be represented in software.
- d) Anonymous objects can be used in establishing default values for parameters.
- e) A single real-world entity may have many different abstractions.

- f) Abstraction may ignore many details about an entity.
- g) Separation refers to "hiding" the implementation of a class in a .cpp file.
- h) An object's destructor is always called when that object goes out of scope (this might be tricky!).
- i) The lifetime of a dynamically created object starts with the use of the new operator and ends when the object goes out of scope. (again, tricky)
- j) Class definitions can be self-referencing.

3. (15 pts) Consider the following code:

```

class Circle {
private:
public:
    Circle();           // create a Circle at Location 10,10 of size 100x100
    SetLocation(Location *newLoc);           // number 1
    SetLocation(Location &newLoc);          // number 2
    SetLocation(int bothXandY = 10);        // number 3
    SetLocation(float scaleFactor);         // number 4
                                           // number 5 none of the above
};
Circle circle;
Location topRight(0, 0);
Location *bottomLeft = new Location(400, 400);

```

Write the number of the method that will be invoked beside each of the following statements:

- a) circle.SetLocation(bottomLeft); _____
- b) circle.SetLocation(topRight); _____
- c) circle.SetLocation(); _____
- d) circle.SetLocation(30); _____
- e) circle.SetLocation(Location(40, 40)); _____
- f) circle.SetLocation(0.5); _____
- g) circle.SetLocation(&topRight); _____
- h) circle.SetLocation(*bottomLeft); _____
- i) circle.SetLocation(new Location(50, 80)); _____
- j) circle.SetLocation(10.5); _____

4. (12 pts) Write the letter of the definition that matches each term below.

<i>Terms</i>	<i>Definitions</i>
_____ 1. Class	a. A named, tangible representation of the attributes and behavior relevant to modeling a given entity for some particular purpose.
_____ 2. Composition	b. A named software representation for an abstraction that separates the implementation of the representation from the interface of the representation.
_____ 3. Abstraction	c. In object-oriented programming, the restriction of access to data within an object to only those methods defined by the object's class
_____ 4. Object	d. A distinct instance of a given class that encapsulates its implementation details and is structurally identical to all other instances of that class.
_____ 5. Association	e. An organized collection of components interacting to achieve a coherent, common behavior.
_____ 6. Encapsulation	f. A type of composition that hides the parts of the composition.
	g. A composition of independently constructed parts that work together.
	h. An organization of abstractions into a directed graph in which the arcs denote an "is-a" relation between a more generalized abstraction and the one or more specializations derived from it.

5. (15 pts) Write the definition of a class “Weight” that could be used as shown below. You need not to show any part of the implementation for the class. Use default arguments as much as possible to minimize the number of methods in your class definition.

```
Weight      Package1 (5, 7);    // Represents a package weighing 5 pounds 7 ounces
Weight      Package2 (9);      // Represents a package weighing 9 pounds 0 ounce
Weight      Package3;          // Represents a package weighing 0 pound 0 ounce

Package1.Add(2, 3);            // Add 2 pounds and 3 ounces of materials to package
Package1.Add(3);               // Add 3 pounds and 0 ounce of materials to package

Package1.Remove(2, 5);         // Remove 2 pounds and 5 ounces of materials
Package1.Remove(2);           // Remove 2 pounds and 0 ounce of materials
Package1.Remove()              // Remove all weight (materials) from the package

int pounds = Package2.Pounds(); // Set pounds to weight of package, rounded to pounds
int ounces = Package2.Ounces(); // Set ounces to weight of package in ounces

int a = Package3.LessThan(Package1); // returns 1 if Package3 is lighter; 0 otherwise
int b = Package3.LessThan(Package2); // returns 1 if Package3 is lighter; 0 otherwise

class Weight {
```

```
};
```

6. (12 pts) Define a **Person** class that contains more than one aggregation and more than one association. Mark "aggregation" by the part of the definition that is the aggregation, similarly for "association." (Just show the header, no code.) Also, state how they are aggregation(s) or association(s).

7. (2 pts) Give the signature of the copy constructor for the Person class.

8. (2 pts) Explain the relationship between an object and a class:

9. (13 pts) What is the output of the following code?

```
#include <iostream.h>
```

```
class M {
private:
    int m;
public:
    M(int x):m(x) {cout << "constructing M: m="<< m << endl; }
    M(const M& copyFrom) {    m = copyFrom.m;
                           cout << "M copy: m=" << m << endl;}
    void operator=(const M& source) { m=source.m;
                                     cout << "M operator=: m=" << m << endl; }
    ~M(){cout << "destructing M" << endl;}
}; // end of class M
```

```
class P {
private:
    M x, y;
public:
    P() : x(5), y(6) {          // larger inline method
                       void someFunc (M param); // just a prototype, does not execute here

                       cout << "constructing P" << endl;
                       x=y;
                       someFunc(x);
                       };
    ~P() { cout << "destructing P" << endl; }
}; // end of class P
```

```
void someFunc(M parm)
{ cout << "in someFunc" << endl; return; }
```

```
void main()
{ P n; }
```

10. (10 pts) Given the following Array definition, give the definition and the implementation for the following overloaded operators. **Include all necessary changes to the Array class.**

```
class Array {
private:
    int array[20];
public:
    Array(int init = 0);           // initializes all array elements to init
    ~Array();
};
```

- a) Write the method definition (signature) and implementation (code) for a + operator. This + operator takes two Arrays and results in an array whose elements are the sum of the elements from the two argument Arrays. For example,
- ```
Array A(2);
Array B(4);
Array C = A + B; // results in all elements of C equal to 6
```

- b) Write a function definition and implementation for a + operator. This + operator takes an int and an Array and adds the value of int to the Array value and returns an Array with that sum. For example,
- ```
Array X(100);
Array Y = 3 + X; // results in all elements of Y equal to 103
```