Instructions:

- Print your name in the space provided below.
- This examination is closed book and closed notes. No calculators or other computing devices may be used.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- Assume any necessary standard header files are incorporated as needed.
- There are 8 parts, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- **Note that failure to return this test, or to discuss its content with a student who has not taken it, is a violation of the Honor Code.**

Do not start the test until instructed to do so!

**Solution**

Name ____________________________

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

__________________________________________

signed
1. Consider the class declaration and implementation:

```cpp
class Foo {
    private:
        int Sz;
        int* A;
    public:
        Foo();
        Foo(int N);
        int Sum();
        ~Foo();
    };

    int Foo::Sum() {
        int tSum = 0;
        for (int i = 0; i < Sz; i++)
            tSum += A[i];
        return tSum;
    }

    Foo::Foo(int N = 0) {
        if (N <= 0) {
            Sz = 0;
            A = NULL;
        } else {
            Sz = N;
            A = new int[N];
            for (int i = 0; i < Sz; i++)
                A[i] = rand() % 100;
        }
    }

    Foo::~Foo() {
        // ????
    }
```

(a) [6 points] A memory leak occurs when a program loses access to dynamically allocated memory without deallocating that memory. Consider the code fragment below. Does a memory leak occur here? If so, describe why and describe how to eliminate the memory leak without changing the logical effect of the code.

```cpp
Foo* p;
for (int k = 0; k < 10; k++) {
    p = new Foo(k);
    cout << p->Sum() << endl;
    p = NULL;
}
```

The first statement in the for loop dynamically allocates a new Foo object, on the system heap, on each pass. That object is never deleted, but access to it is lost when the last statement in the for loop is executed. That’s a classic instance of a memory leak. The leak can be eliminated by changing the final statement in the for loop to:

```cpp
delete p;
```

Alternatively, insert the delete statement prior to the last statement of the for loop.

(b) [6 points] Does the class Foo need a destructor? Why or why not? If so, write a correct implementation of the destructor.

Yes. A Foo object contains a pointer to memory that is allocated dynamically when the object is constructed. That memory will NOT be deallocated properly unless the Foo destructor does so:

```cpp
Foo::~Foo() {
    delete [] A;
}
```
2. [16 points] Consider the following class declaration and implementation:

```cpp
class Point {
    private:
        int X, Y;
    public:
        Point() {X = 0; Y = 0;};
        Point(int xcoord, int ycoord) { X = xcoord; Y = ycoord;};
        int getX() const {return X;};
        int getY() const {return Y;};
        void setX(int xcoord) {X = xcoord;};
        void setY(int ycoord) {Y = ycoord;};
};
```

(a) Given the array declaration below, what are the values of the data members of `Hexagon[2]`?

```cpp
Point Hexagon[6];
```

When an array of objects is declared, the default class constructor is applied to each array cell, so each element of the array `Hexagon` will store the data values 0 and 0.

(b) Given the function `yFlip()` shown at right, what value is printed by the code below?

```cpp
Point C(43, 29);
cout << C.getX() << endl;
```

Should have included a call to `yFlip()`; as it is, it will obviously print the value 29.

(c) What value is printed by the code below?

```cpp
Point A(17, -24);
Point B = A;
cout << B.getY() << endl;
```

The assignment just copies the data member values in `A` to the corresponding members of `B`, so this will print –24.

(d) Complete the implementation of the following member operator:

```cpp
bool Point::operator==(const Point& Other) {
    return ( (X == Other.X) &&
            (Y == Other.Y));
}
```

```cpp
void yFlip(Point P) {
    P.setX(-P.getX());
}
```
3. Consider the partial class declarations given below:

```cpp
class A {
    private:
        int Count;
        char Flag;
    public:
        A(int C=0, char F = 'x');
        int getCount() const;
        char getFlag() const;
};

class B {
    private:
        A* myA;
    public:
        B(int Sz = 5) {myA = new A[Sz]}
        ~B() {delete [] myA;}
};

class C {
    private:
        A* myA;
    public:
        C(const A& X) {myA = &X}
        ~C() {myA = NULL;}
};
```

(a) [5 points] Is the relationship between class B and class A one of aggregation or association or neither? Why?

When an object of class B is created, an array of objects of class A is created; that array is then destroyed when the object of class B is destructed. Since the A’s have no existence independent of that of the object of class B, this is an aggregation, not an association.

(b) [5 points] Is the relationship between class C and class A one of aggregation or association or neither? Why?

In contrast to (a), here an object of class C stores the address of an object of class A, but the instance of A is NOT created by C’s constructor, nor is it deleted by C’s destructor. Here the instance of A does have an independent existence, so this is an association, not an aggregation.

(c) [5 points] Is the relationship between class B and class C one of aggregation or association or neither? Why?

There is NO relationship between the classes B and C.
4. Consider the classes declared below:

```cpp
class M {
private:
    int m;
public:
    M(int x) : m(x) {
        cout << "Constructing M(" << x << ")" << endl;
    }
    int getData() const { return m; }
    ~M() { cout << "Destructing M" << endl; }
};
class Z {
private:
    M a;
public:
    Z() {
        a = 0;
        cout << "Constructing Z(" << a.getData() << ")" << endl;
    }
    Z(int m) : a(m) {
        cout << "Constructing Z(" << a.getData() << ")" << endl;
    }
};
```

(a) [4 points] What output, if any, would be produced by the following declaration?

```
M M1(13);
```

This will simply fire the constructor for M, producing: **Constructing M(13)**

(b) [4 points] What output, if any, would be produced by the following declaration?

```
Z Z1(5);
```

Since a Z contains an M, this will fire both constructors; the “inside” objects are constructed first, so:

**Constructing M(5)**
**Constructing Z(5)**

(c) [4 points] What output, if any, would be produced when the object Z1 declared above is destructed?

Again, since a Z contains an M, both destructors will fire. Destruction is done from the outside-in, so the Z is destructed (producing NO output) and then the M:

**Destructing M**
5. Consider the classes Track and Album:

```cpp
class Album {
private:
    string Title;
    string Artist;
    Track* PlayList;
public:
    Album(string T, string A, int numTracks);
    bool AddTrack(const Track& T);
    Track getTrack(int Position) const;
    ~Album();
};
Album::Album(string T, string A, int numTracks) {
    Title = T;
    Artist = A;
    PlayList = new Track[numTracks];
}
...
Album::~Album() {
    delete [] PlayList;
}

class Track {
private:
    string Title;
    int Length;
public:
    Track(string T, int L);
    string getTitle() const;
    int getLength() const;
};
```

Suppose a function is implemented to read Track data from an input file and add corresponding Track objects to the array:

```cpp
void initTracks(Album CD, ostream& In) {
    // Correct code to read data, create Track objects, and
    // add them to CD by calling CD.AddTrack().
}
```

(a) [10 points] Calling the function `initTracks()` will have an unfortunate side effect (even though the body of the function is correct). Describe that side effect clearly.

(b) [6 points] Describe clearly how the side effect can be eliminated without altering the implementation of `initTracks()`. Do not write C++ code to answer the question.

This question was flawed and was not counted. See the last page for the substitute question that was given as an in-class quiz to replace this one.
6. Consider the template \texttt{StackT}: 

```cpp
template <class Foo, int Size> class StackT {
private:
    Foo S[Size];
    int Top;
public:
    StackT();
    bool Push(Foo Item);
    bool Pop(Foo& Item);
    bool isEmpty() const;
    bool isFull() const;
    ~StackT();
};
```

[6 points] Determine which of the following declarations are valid (will compile):

(a) \texttt{StackT\langle\text{double}, 1000\rangle Q1; valid invalid}

(b) \texttt{const int Size = 42; StackT\langle\text{double}, Size\rangle Q1; valid invalid}

(c) \texttt{int Size; cin >> Size; StackT\langle\text{double}, Size\rangle Q1; valid invalid}

[6 points] Given the following valid declarations, determine which of the following assignments are valid (will compile):

\begin{align*}
\texttt{StackT\langle\text{string}, 100\rangle S1;} \\
\texttt{StackT\langle\text{string}, 100\rangle S2;} \\
\texttt{StackT\langle\text{string}, 500\rangle S3;} \\
\texttt{S2 = S1;} & \text{ valid invalid} \\
\texttt{S3 = S1;} & \text{ valid invalid}
\end{align*}

7. [2 points] Choose the best answer. The use of public data members in a class is:

(a) sound software engineering practice.

(b) acceptable software engineering practice.

(c) an abomination in the eyes of man and God.
8. Consider the following class diagram for the Library System from Project 2:

From the discussions in class, one should consider the behavioral perspective when determining the classes that will be used, and how they will interact. The behavioral perspective considers what actions must be supported by the implementation and what each action implies about the classes and their relationships.

(a) [3 points] Clearly describe one action that must be supported in the Library System.

(b) [3 points] Considering the action identified in (a), and the exact class diagram given above, which object (class) will have the responsibility of initiating the action?

(c) [3 points] Considering the action identified in (a), and the exact class diagram given above, which object(s) (classes) will collaborate to perform the action?

(d) [3 points] Considering the action identified in (a), and the exact class diagram given above, which object(s) (classes) will be altered by the performing of the action?

(e) [3 points] Considering the action identified in (a), and the exact class diagram given above, which object(s) (classes) will be interrogated (i.e., accessed for state information) during the performing of the action?
Consider the classes Track and Album:

```cpp
class Album {
    private:
        string Title;
        string Artist;
        int numTracks;
        Track* PlayList;
    public:
        Album(string T, string A, int numTracks);
        bool AddTrack(const Track& T);
        Track getTrack(int Position) const;
        ~Album();
    };  
Album::Album(string T, string A, int nT) {
    Title = T;
    Artist = A;
    numTracks = nT;
    PlayList = new Track[nT];
}
.
.
Album::~Album() {
    delete[] PlayList;
}
```

Suppose a function is implemented to compute the total length of all the Tracks on an album (in seconds):

```cpp
int Length(Album CD) {
    // Correct code to compute and return the total length
    // of all the tracks.
}
```

(a) [10 points] Calling the function Length() will have an unfortunate side effect (even though the body of the function is correct). Describe that side effect clearly.

The parameter to Length() is passed by value; however, an Album object contains a pointer to a dynamically allocated array of Track objects, and Album does not implement a copy constructor to provide a deep copy. Therefore, the actual parameter (used in the call) and the formal parameter CD will share the same array.

When Length() terminates, the local variable CD will be destructed. That will delete the array of Track objects that CD shares with the actual parameter; so the actual parameter will have been modified (corrupted actually) even though it was passed by value.

(b) [6 points] Describe clearly how the side effect can be eliminated without altering the implementation of Length(). Do not write C++ code to answer the question.

Implement a deep copy constructor for Album.

Passing the parameter by constant reference is a modification of Length(); passing the parameter by reference is simply unacceptable SE practice.