Instructions:

- Print your name in the space provided below.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that you have done so, and label the continuation with the question number.
- If you want partial credit, justify your answers briefly and concisely, even when justification is not explicitly required.
- There are 11 questions, 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.
- This is a closed-book, closed-notes examination. No calculators or other electronic devices may be used during this examination. You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it. You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.

Do not start the test until instructed to do so!

Name ___________________________ printed

PID ___________________________ printed

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

_____________________________ signed
For the next five questions, consider the partial class code that follows:

```
#include "Thing.h"
#include "Patron.h"

class Box {
    private:
        Thing** store;
        int dim, num;
    public:
        Box(int cap = 3);
        bool Add(Thing* const tp);
        bool Sub(Thing* const tp);
        ~Box();
        friend ostream& operator<<(ostream& out, const Box& bx);
    }

Box::Box(int cap) {
    dim = cap; num = 0;
    store = new Thing*[dim];
}

Box::~Box() {
    delete [] store;
}

bool Box::Add(Thing* const tp) {
    if (num == dim) return false;
    store[num++] = tp;
    return true;
}

bool Box::Sub(Thing* const tp) {
    for (int i=0; i<num; i++)
        if (store[i] == tp) {
            for (int j=i+1; j<num; j++)
                store[j-1] = store[j];
            num--;
            return true;
        }
    return false;
}

ostream& operator<<(ostream& out, const Box& bx) {
    for (int i=0; i<bx.num; i++)
        out << *(bx.store[i]) << endl;
    return out;
}

class Club {
    private:
        Patron* members;
        int num;
    public:
        Club(istream& membs = cin, int size);
        void List(ostream& out);
        ~Club();
    }

Club::Club(istream& membs = cin, int size) {
    string pat;
    Patron p;
    num = size;
    members = new Patron[num];
    for (int i=0; i<num; i++) {
        getline(membs, pat);
        members[i] = p.parse(pat);
    } // parses Patron records
}

Club::~Club() {
    delete [] members;
}

class Manage {
    private:
        Box* B;
        Club* C;
    public:
        Manage(istream& membs, int size);
        void Execute();
        ~Manage();
    }

Manage::Manage(istream& membs, int size) {
    B = new Box(size);
    C = new Club(membs, size);
}

void Execute() {
    // omitted code
}

Manage::~Manage() {
    delete B; delete C;
}
```
1. [20 points] Assume that the Thing and Patron classes referred to in the previous code are simple data classes, containing only primitive variables and the insertion operator << is overloaded in them. Draw the class relationship diagram, with multiplicity values, depicting all of the associative and aggregation class relationships:

2. [5 points] Consider execution of the following client code fragment:

```c++
Box* B;                        // Line 1
for (int i = 0; i < 3; i++) {  //      2
    B = new Box[3];            //      3
    delete [] B;               //      4
}
```

Determine whether this code causes a runtime error and/or a memory leak. If yes, explain clearly how the leak occurs. If no, explain clearly, referring to line numbers, what prevents a leak from occurring.
3. [10 points] Assume the constructors and destructors in the code for question one were instrumented to display a simple message when they were executed. Also assume that the Thing class is instrumented. Give a brief description of the output that would result due to the correct execution of the code segment in the previous question. (Do not give a listing of the instrumented messages; just describe the groups of messages that would be output. In other words, explain in what order which constructors and destructors would fire.)

4. [5 points] Consider execution of the following client code fragment:

```cpp
Club* C;
for (int i = 0; i < 3; i++) {
    C = new Club[1];
    Display( C[0] );
    delete [] C;
}

void Display(Club C) {
    C.List(cout);
}
```

Determine whether this code causes a runtime error and/or a memory leak. If yes, explain clearly how the runtime error and/or leak occurs.

5. [8 points] What obvious methods are missing from the Box and Club classes in order for them to operate correctly?
#include <iostream>
using namespace std;

class X {
public:
   X() {cout << "constructing X" << endl;}
   virtual ~X() {cout << "destructing X" << endl;}
};
class Y : public X {
public:
   Y() {cout << "constructing Y" << endl;}
   virtual ~Y() {cout << "destructing Y" << endl;}
};
class Z : public Y {
public:
   Z() {cout << "constructing Z" << endl;}
   virtual ~Z() {cout << "destructing Z" << endl;}
};

int main() {
   X *x = new Z();
   delete x;
   return 0;
}

6. [6 points] Give the output, if any, the program above produces.
For the next 4 questions, consider the following class hierarchy and program:

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

class A {
private:
  int a;
public:
  A(int i = 1): a(i) {} //{a = i;}
      int getA() const {return a;}
};

class B {
private:
  A   ab;
  int bb;
public:
  B(A a2 = A(2), int b = 2): ab(a2), bb(b) {} //{ab = a2; bb = b;}
      int getBB() const {return bb;}
      A   getA2() const {return ab;}
};

class D : public B {
private:
  int ddd;
public:
  D(int d = 3): ddd(d) {} //{ddd = d;}
      D(B b, int d): B(b), ddd(d) {}
      D(A a, B b, int d): B(a, b.getBB()), ddd(d) {}
      int getD() const {return ddd;}
      int Sum() const; //see below for possible implementation
};
```

For the next two questions, consider the following declaration:

```cpp
D d( A(1), B( A(2), 3), 4);
```

7. [5 points] Can one make the following member invocation, (very briefly explain)? (If yes, give the output.)

```cpp
int i8 = d.getA();
cout << "i8 = " << i8 << endl;
```

8. [5 points] Can one make the following member invocation, (very briefly explain)? (If yes, give the output.)

```cpp
int i9 = d.getBB();
cout << "i9 = " << i9 << endl;
```
For the next question, consider the following implementation of D::Sum():

    int D::Sum() const {
        return (ddd + bb + getA2().getA());
    }

9. [6 points] Briefly explain why this implementation is valid or invalid. If you believe the implementation is invalid briefly indicate, (without eliminating inheritance), what simple change(s) would make this implementation valid.

10. [15 points] A marina at a small local lake rents various types of watercraft. You have been hired to develop a software system to help them keep track of their rental fleet. Their rental stock consists of the following various types of watercraft:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PaddleBoat</td>
<td>Small pontoon float craft, seats two. Powered by leg cycling to turn a paddle wheel.</td>
</tr>
<tr>
<td>Ski Boat</td>
<td>Inboard/outboard 150hp motorized, fiberglass craft, seating up to 5 people.</td>
</tr>
<tr>
<td>Houseboat</td>
<td>Outboard 75hp motorized pontoon craft for water camping, sleeps up to 7 persons.</td>
</tr>
<tr>
<td>Canoe</td>
<td>Light, narrow, aluminum, cylindrical craft, with two paddles, holding up to 3 people.</td>
</tr>
<tr>
<td>Fishing Boat</td>
<td>Outboard 90hp motorized, fiberglass craft, with 3 swivel seats, livewell and baitwell.</td>
</tr>
<tr>
<td>Pontoon Boat</td>
<td>Outboard 50hp motorized pontoon craft for pleasure cruising, (no sleeping facilities)</td>
</tr>
<tr>
<td>Rowboat</td>
<td>Small aluminum, V-hull craft, with two oars, holding up to 5 people.</td>
</tr>
<tr>
<td>Barge</td>
<td>A rectangular, towable raft, used for hauling.</td>
</tr>
</tbody>
</table>

In order to adopt an optimal design, you have decided that you need to develop a possible inheritance hierarchy for comparison to alternative designs. Considering only the watercraft types listed above, draw a sensible inheritance hierarchy showing only the classes. You do not have to show any class members. (Hint: consider the information above and much less about the program being developed.)
For the next question, consider the following class hierarchy and program:

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

class One {
public:
    virtual int Fn1() { return(-1); }
    virtual int Fn2() = 0;
    virtual int Fn3() { return(1); }
    int Fn4() { return(2); }
};
class Two : public One {
public:
    int Fn2() { return(3); }
    int Fn3() { return(4); }
    virtual int Fn5() { return(7 + Fn1()); }
};
class Three : public Two {
public:
    int Fn1() { return(5); }
    int Fn3() { return(6); }
};

int main() {
    Three *t3 = new Three();
    Two   *t2 = t3;
    One   *t1 = t2;
    
    cout << t1->Fn2() << endl;         // Line 0
    cout << t1->Fn3() << endl;         // Line 1
    cout << t1->Fn5() << endl;         // Line 2
    cout << t2->Fn4() << endl;         // Line 3
    cout << t2->Fn5() << endl;         // Line 4
}
```

11. [15 points] Determine ALL output, if any, the indicated line produces. Consider each statement separately. If a statement will not compile or result in an execution error very briefly explain why.

<table>
<thead>
<tr>
<th>Line</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>