Instructions:

- Print your name in the space provided below.
- Answer each question in the space provided.
- If you want partial credit, justify your answers briefly and concisely, even when justification is not explicitly required.
- There are 12 questions, priced as marked. The maximum score is 100.
- Legibility counts. Responses that cannot be easily read may not be graded!
- 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 a Va Tech Honor Code violation, (www.honorsystem.vt.edu).

Do not start the test until instructed to do so!

Circle your instructor and section:

Barnette     Wheaton
8:00AM TuTh  11:00AM TuTh  10:10AM MWF  12:20PM MWF

Name ____________________________ printed

PID ____________________________ VT email address (NOT SSN) - printed

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

______________________________
Signed

Spring 2004
1. [6 points] Consider the following: If A and B are objects from two classes, what relationship type is described by each of the following statements?

   A has-a B   Aggregation

   A knows-a B   Association

   A is-a-kind-of B   Inheritance

2. [5 points] In order to implement late (execution time) binding of function invocations to specific function definitions, C++ compilers create virtual function tables for classes in a polymorphic inheritance hierarchy. Briefly (very), explain how the virtual function table for a class is located during execution, (NOT how binding is decided)?

   The presence of a virtual function in a class causes the generation of a virtual function table (vtbl) for that class, and an association to the vtbl in each object of that type. At runtime, the association link in the object is used to access the virtual function table for its class.

For the next two questions, consider the following simple class for a Checking Account. Note that many of the member functions that are not relevant are omitted.

```cpp
//  "CheckingAccount.h"
class CheckingAccount {
private:
    Person *owner;
    ID accountNum;
    Balance balance;

public:
    CheckingAccount();
    CheckingAccount(Person *name, 
        ID acctNum=ID(0), 
        Balance amount = Balance(0.0));

    // possible missing function
    // prototypes?
~CheckingAccount();
};

//  "CheckingAccount.cpp"
#include "CheckingAccount.h"

CheckingAccount::CheckingAccount( ) : 
    owner(NULL),
    accountNum(0), balance(0.0) { }

CheckingAccount::CheckingAccount( 
    Person *name, 
    ID acctNum, Balance amount) 
{ 
    owner = name;
    accountNum = acctNum;
    balance = amount;
}

//  possible missing functions?
CheckingAccount::~CheckingAccount() { }
```
3. [6 points] Which data members in the `CheckingAccount` class are examples of association? Which are examples of aggregation? Briefly explain your answers.

- **owner** – association (static) The Person object can exist before & after the CheckingAccount object.
- **accountNum** – aggregation (static) The accountNum & balance objects are created and destroyed with the balance – aggregation CheckingAccount object.

4. [12 points] Next suppose that the `CheckingAccount` constructors and destructor are instrumented (i.e. they contain a print statement indicating that a CheckingAccount object has been created/destroyed). Also, assume the constructor(s) and destructor(s) for all other classes from problem 3 also have a similar print statement (i.e. they are also instrumented). NOTE that the copy constructors are NOT implemented and thus not instrumented, (i.e., copy construction will NOT produce any output). Given the following simple program, what is the expected output? (You may wish to label which line causes each line of the output.)

```cpp
void main() {
    Person P;       //Line 0
    ID i(5);        //Line 1
    CheckingAccount C(&P, i);  //Line 2
}                   //Line 4
```

```
Constructing (default) Person  //1
Constructing ID                //2
Constructing Balance           //3 construction of the anonymous default argument
Constructing (default) ID     //3 aggregated construction
Constructing (default) Balance //3 aggregated construction
Constructing CheckingAccount  //3 construction of containing object
Destructing ID                //3 destruction of value parameter
Destructing Balance           //3 destruction of value parameter
Destructing Balance           //3 destruction of anonymous default argument
Destructing CheckingAccount   //4 out of scope destruction
Destructing Balance           //4 out of scope destruction
Destructing ID                //4 out of scope destruction
Destructing ID                //4 out of scope destruction
Destructing Person            //4 out of scope destruction
Press any key to continue
```
For the next two questions, consider the following simple class hierarchy:

```cpp
// Baseball Player
class BballPlayer {
private:
    string name; // Player's Name
    float avg; // Batting Average
public:
    BballPlayer(string pname="", float bavg=0.0) : name(pname), avg(bavg) { }
    void SetAvg(float bavg) { avg=bavg; }
    float GetAvg() const { return(avg); }
    string GetName() const { return(name); }
    void Display() const { cout << name << " has a batting average of " << avg; }
};

class Pro : public BballPlayer {
private:
    int salary; // Annual salary in dollars
public:
    Pro(string pname="", float bavg=0.0, int dollars=0); // Missing Code...
    void SetSalary(int dollars) { salary=dollars; }
    int GetSalary() const { return(salary); }
    void Display() const { cout << name << " has a batting average of " << avg << " and a salary of " << salary; }
};
```

5. [5 points] Note that the code for the constructor of the `Pro` class is missing. Provide a possible implementation for the constructor.

```cpp
Pro(string pname, float bavg, int dollars) :
    BballPlayer(pname, bavg), salary(dollars) { }
```
6. [10 points] One of the member functions in the above class code contains a common error dealing with inheritance. Specify the function that contains the error, briefly explain the error, and then give the code for two simple fixes for the problem.

Pro::Display – name and avg are not directly accessible since they are private data members of BballPlayer.

1. Change name and avg from private to protected
2. Change Pro::Display to:
   ```
   {
     BballPlayer::Display(); //invoke base class fn
     cout << salary;
   }
   ```

Other possibilities?

3. Change name and avg from private to public (Yuk!)
4. Make Pro a friend class of BballPlayer
5. Use the base accessor methods to display the inherited members
For the next three questions, consider the following poorly designed class hierarchy:

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

class Access { //Net access method (http, ftp, mailto, …)
private:
    string p; // protocol
public:
    Access(string a = "http://") : p(a) {}
        // { p = a; }
    string prot() const { return(p); }
    virtual string waddr() const = 0; //web address
};

class Domain : public Access { //machine net name
private:
    string d; // "local.subdomain.domain"
public:
    Domain(string dom = "www.w3.org") : d(dom) {}
        // { d = dom; }
    string locale() const { return(d); }
    string waddr() const { return(prot() + d); }
    string machine() const { return(d.substr(0, d.find("."))); }
};

class Info : public Domain { //file web information
private:
    string i;
public:
    Info(string in = "/index.html") : i(in) {}
        // { i = in; }
    string data() const { return(i); }
    string waddr() const { return(prot() + locale() + i); }
    virtual string type() const {
        return(i.substr(i.find("."), i.length()-i.find(".")-1));
    }
};
```

For the next three questions, consider the following client code declarations:

```
Domain d1("www.vt.edu");
Domain* d2 = &d1;
Info* f1 = new Info("default.htm");
Domain* d3 = f1;
Access* a1 = d2;
```

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

```
string s8 = d3->type();
cout << "s8 = " << s8 << endl;
```

No, there is no `type()` member in `Domain` or its base class `Access`.

The compiler will not even implement polymorphism/late binding during compilation if the class of the pointer type (or its base class) does not contain a virtual function prototype declaration.
8. [7 points] Can one make the following member invocation, (very briefly explain)? (If yes, also give the output.)

```cpp
string s9 = f1->machine();
cout << "s9 = " << s9 << endl;
```

Yes, Info inherits (not virtually) `machine()` from Domain.

**Output**

`s9 = www`

9. [7 points] Can one make the following member invocation, (very briefly explain)? (If yes, also give the output.)

```cpp
string s10 = a1->waddr();
cout << "s10 = " << s10 << endl;
```

Yes, the Access base class contains a pure virtual `waddr()` that is over-rode in Domain.

**Output**

`s10 = http://www.vt.edu`
For the next 2 questions, consider the following class hierarchy:

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

class A {
public:
    virtual void one() = 0;
    virtual void two() {cout << "A::two()" << endl;}
    void three() {cout << "A::three()" << endl;}
    virtual void four() = 0;
};
class B : public A {
public:
    void one() {cout << "B::one()" << endl;}
    void two() {cout << "B::two()" << endl;}
    virtual void three() {cout << "B::three()" << endl;}
};
class C : public B {
public:
    void three() {cout << "C::three()" << endl;}
    void four() {cout << "C::four()" << endl;}
};
```

Given the following virtual table for class A:

<table>
<thead>
<tr>
<th>Function identifier</th>
<th>Function pointer</th>
<th>Bound Function (Class :: Function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>one()</td>
<td>→</td>
<td>NULL</td>
</tr>
<tr>
<td>two()</td>
<td>→</td>
<td>A :: two()</td>
</tr>
<tr>
<td>four()</td>
<td>→</td>
<td>NULL</td>
</tr>
</tbody>
</table>

10. [10 points] Give the Virtual table for class B:

<table>
<thead>
<tr>
<th>Function identifier</th>
<th>Function pointer</th>
<th>Bound Function (Class :: Function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>one()</td>
<td>→</td>
<td>B :: one()</td>
</tr>
<tr>
<td>two()</td>
<td>→</td>
<td>B :: two()</td>
</tr>
<tr>
<td>three()</td>
<td>→</td>
<td>B :: three()</td>
</tr>
<tr>
<td>four()</td>
<td>→</td>
<td>NULL</td>
</tr>
</tbody>
</table>
11. [10 points] Give the Virtual table for class C:

<table>
<thead>
<tr>
<th>Function identifier</th>
<th>Function pointer</th>
<th>Bound Function (Class :: Function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>one()</td>
<td>→</td>
<td>B :: one()</td>
</tr>
<tr>
<td>two()</td>
<td>→</td>
<td>B :: two()</td>
</tr>
<tr>
<td>three()</td>
<td>→</td>
<td>C :: three()</td>
</tr>
<tr>
<td>four()</td>
<td>→</td>
<td>C :: four()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
You have been selected to develop an inheritance hierarchy for various types of aircraft in order to minimize the code size for the system. The following various types of aircraft and data must be modeled:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>747</td>
<td>Large 4-engine passenger jet</td>
</tr>
<tr>
<td>Blimp</td>
<td>Non-rigid airship typically filled with helium</td>
</tr>
<tr>
<td>C-130</td>
<td>4-engine military airplane that is used to carry troops and cargo</td>
</tr>
<tr>
<td>Hot-air Balloon</td>
<td>Airship that consists of a basket and an air-filled balloon that is heated by an adjustable flame</td>
</tr>
<tr>
<td>Concorde</td>
<td>Supersonic passenger jet typically used for transcontinental flights</td>
</tr>
<tr>
<td>B-2</td>
<td>Military jet that is used for bombing and is designed to avoid radar detection</td>
</tr>
<tr>
<td>Turbo-prop</td>
<td>Turbo powered airplane that is used by private pilots</td>
</tr>
<tr>
<td>Air Force 1</td>
<td>Large 4-engine passenger jet that is customized for the President of the United States</td>
</tr>
<tr>
<td>Bi-plane</td>
<td>Single-engine fighter airplane that was used during World War I</td>
</tr>
</tbody>
</table>

12. [15 points] Considering only the possible types listed above, draw a sensible class relationship/inheritance hierarchy. You do not have to show any class members. (Hint: consider the information above and much less about any program being developed from the hierarchy. Be aware that base/intermediate types not explicitly listed above may need to be modeled for organizational purposes.)
Alternative Design:

![Alternative Design Diagram]

Aircraft
- Engine-Powered
  - Jets
    - Military
      - B-2
    - Civilian
      - Concord
    - Bi-plane
  - Props
    - Turbo-prop
    - C-130
- Airship
  - Hot-air Balloon
  - Blimp
- Air Force 1

747