Name: _____

CSE 303, Autumn 2006, Final Examination

12 December 2006

`Solutions`

Please do not turn the page until everyone is ready.

Rules:

- The exam is closed-book, closed-note, except for one two-sided 8.5"x11" piece of paper.
- Please stop promptly at 4:20.
- You can rip apart the pages, but please write your name on each page.
- There are **79 points** total, distributed **unevenly** among 6 questions (many of which have multiple parts).
- When writing code, style matters, but don't worry about indentation.

Question	Max	Grade
1	11	
2	16	
3	10	
4	13	
5	9	
6	20	
Total	79	

Advice:

- Read questions carefully. Understand a question before you start writing.
- **Write down thoughts and intermediate steps so you can get partial credit.**
- The questions are not necessarily in order of difficulty. **Skip around.**
- If you have questions, ask.
- Relax. You are here to learn.

Name: _____

1. (**11 Points**) Consider the following three header/source files:

```
--- bar.h ------------------------------------------------

struct Bar {
  int baz;
  int quux;
};

--- foo.h ------------------------------------------------

#include "bar.h"
int foo(struct Bar *bar);

--- foo.c ------------------------------------------------

#include <stdlib.h>
#include "foo.h"
#include "bar.h"

int main(int argc, char**argv) {
  struct Bar b = {atoi(argv[1]), atoi(argv[2])};
  return foo(&b);
}

int foo(struct Bar *bar) {
  return (bar->baz == bar->quux) ? 1 : 0;
}
```

(a) (2 pts) What error will you get when you compile with `gcc –Wall –o foo foo.c`?

struct Bar is defined twice. The actual error is as follows:

```
In file included from foo.h:1,
                 from foo.c:2:
bar.h:1: error: redefinition of struct Bar
```

(b) (2 pts) How can you change `foo.h` so that compilation will succeed? Be specific.

Replace the following line…

```
     #include “bar.h”
```

with this line…

```
     struct Bar;
```

This is a “forward declaration” and does not include the definition, so the compile succeeds.

Name: _____

Problem 1, continued

(c) (2 pts) How can you change `bar.h` so that compilation will succeed? Be specific.

Add these two lines to top…
```
#ifndef BAR_H
#define BAR_H
```
…and add this line to bottom:
```
#endif
```

This makes it so the header file will be included at most once.

(d) (2 pts) Is it best to fix this problem by changing `foo.h`, `bar.h`, or both? Why?

Both. Fixing bar.h allows it to be included anywhere without a problem, which should reduce future compilation troubles. Fixing foo.h reduces dependencies, because files that include foo.h do not automatically include bar.h. Reduced dependencies can lead to faster recompilation in larger projects.

(e) (3 pts) What will happen when you run this program in the following ways? Give the code that is returned and/or describe any errors that occur.

- `./foo 1 1`

  Return code 1

- `./foo 1 One`

  Return code 0

- `./foo 1`

  Acceptable Answers:
  - **Array out of bounds**
  - **Unpredictable**
  - **Segmentation Fault**

Name: _____

Name: _____

2. (**16 Points**) Consider the following Class definitions. (Feel free to rip out this page.)

```
class A {

  char *str;

public:
  A(char *s) : str(s) {}

  ~A() {}

  char *getStr() { return str; }
};


class B {
  char *str;

public:
  B(char *s) {
    str = (char *)malloc(strlen(s) + 1);
    strcpy(str,s);
  }

  ~B() { free(str); }

  char *getStr() {
    char *newStr = (char *)malloc(strlen(str) + 1);
    strcpy(newStr,str);
    return newStr;
  }
};
```

Name: _____

Name: _____

Problem 2, continued

(a) (2 pts) These two classes have the same data members and the same interface. Why is class `B` more robust?

Acceptable answers:
- **Because it has better encapsulation/information hiding.**
- **Internal data is copied before being stored or returned.**
- **The internal data member is not accessible to callers.**

(Partial credit will be given if "copying" is mentioned.)

(b) (2 pts) What information should be included in a minimum specification for `B::B(char *s)`?

1) The string s is copied.
2) Precondition: s cannot be NULL

(c) (2 pts) What information should be included in a minimum specification for `B::getStr()`?

1) The method returns a copy of the string in the object.
2) The caller is responsible for freeing the memory.

(d) (2 pts) Write a line of code that you could add to the top of `B::B(char *s)` to make it more robust.

```
assert(s != NULL);
```

Name: _____

Problem 2, continued

For each of the following code sequences, say what compile time or run time problems it causes, if any.

(e) (2 pts)

```
void f1() {
  A a("A");
  B b("B");
  b = (B)a;
}
```

Compile error: A constructor for B that takes A does not exist.

(f) (2 pts)

```
void f2() {
  A *a = (A*)new B("B");
  delete a;
}
```

No compile error: The cast is not checked.
Memory leak: A's destructor will be invoked, which does not free the string copy made by B.

(g) (2 pts)

```
void f3() {
  A *a = dynamic_cast<A*>(new B("B"));
  delete a;
}
```

Compile error: dynamic_cast<A*> will only take arguments that are pointers or references subclasses or superclasses of A.

(h) (2 pts)

```
void f4() {
  B b1("B1"), b2("B2");
  b2 = b1;
}
```

Run time error: Crash: The automatically created operator= will copy b1.str to b2.str, and at the end of the function, that pointer will be freed twice. Also leaks memory (copy of string "B2").

Name: _____

Continue to the next page

Name: _____

3. (**10 Points**) Consider the following code:

```
#include <iostream>

using namespace std;


class C1 {
  int _i1;

public:
  C1(int i1) : _i1(i1) {}

  virtual void PrintA(ostream &out) {
    out << _i1 << endl;
  }
  void PrintB(ostream &out) {
    out << _i1 << endl;
  }
};


class C2 : public C1 {
  int _i2;

public:
  C2(int i1, int i2) : C1(i1), _i2(i2) {}

  void PrintA(ostream &out) {
    out << _i2 << endl;
  }
  virtual void PrintB(ostream &out) {
    out << _i2 << endl;
  }
};


class C3 : public C2 {
  int _i3;

public:
  C3(int i1, int i2, int i3) : C2(i1,i2), _i3(i3) {}

  void PrintA(ostream &out) {
    out << _i3 << endl;
  }
  void PrintB(ostream &out) {
    out << _i3 << endl;
  }
};
```

Name: _____

Problem 4 (continued)

For each commented line below, indicate what is printed on cout.

```
int main() {
  C1 c1(1);
  C2 c2(1,2);
  C3 c3(1,2,3);

  C1 *p1 = &c1;
  C1 *p2 = dynamic_cast<C1*>(&c2);
  C1 *p3 = dynamic_cast<C1*>(&c3);

  p1->PrintA(cout);   // Output is _________  Ans: 1

  p1->PrintB(cout);   // Output is _________  Ans: 1

  p2->PrintA(cout);   // Output is _________  Ans: 2

  p2->PrintB(cout);   // Output is _________  Ans: 1

  p3->PrintA(cout);   // Output is _________  Ans: 3

  p3->PrintB(cout);   // Output is _________  Ans: 1


  C1 c1_2 = c2;
  C1 c1_3 = c3;


  c1_2.PrintA(cout);   // Output is _________  Ans: 1

  c1_2.PrintB(cout);   // Output is _________  Ans: 1

  c1_3.PrintA(cout);   // Output is _________  Ans: 1

  c1_3.PrintB(cout);   // Output is _________  Ans: 1

}
```

Name: _____

4. (**13 Points**) The following is a list of files and the include statements found in each:

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

-- Pilot.cpp ------------------------------
#include "Pilot.h"

-- Viper.h --------------------------------
#include <iostream>

-- Viper.cpp ------------------------------
#include <cassert>
#include "Viper.h"
#include "Battlestar.h"
#include "Pilot.h"

-- Battlestar.h ---------------------------
#include <iostream>
#include <string>
#include <vector>

-- Battlestar.cpp -------------------------
#include <cassert>
#include "Battlestar.h"
#include "Viper.h"

-- main.cpp -------------------------------
#include <iostream>
#include <iostream>
#include "Battlestar.h"
#include "Viper.h"
```

Suppose we also have a `Makefile` that begins with the following lines.

```
-- Makefile -------------------------------

GXX = g++
CFLAGS = -Wall –g
```

Name: _____

Instead of using static libraries, we want this `Makefile` to compile `main.cpp`, `Battlestar.cpp`, `Viper.cpp`, and `Pilot.cpp` and link them to create a program called `bsgsim`. Indicate whether or not the following lines of the Makefile are correct. If they are incorrect, write the corrected version.

(a) (3pts)

```
Battlestar.o: Battlestar.cpp Battlestar.h Viper.h
      g++ -Wall -g -c -o $@ $<
```

Correct

(b) (3pts)

```
Pilot.cpp: Pilot.h
      g++ -Wall -g -c –o Pilot.o $<
```

Corrected Version:

```
Pilot.o: Pilot.cpp Pilot.h
      g++ -Wall -g -c -o Pilot.o $<
```

(c) (3pts)

```
bsgsim: main.o Viper.o Battlestar.o
      $(GXX) $(CFLAGS) -o $@ $<
```

Corrected Version:

```
bsgsim: main.o Viper.o Battlestar.o
      $(GXX) $(CFLAGS) -o $@ $^
```

For each statement below, indicate whether it is true or false.

(d) (1 pt) In order for the whole project to compile when we type "`make`", the line that builds "`bsgsim`" should go at the bottom of the `Makefile`. **Ans: False**

True False

(e) (1 pt) In order to run `bsgsim` in `gdb`, we would have to add a special flag to each `Makefile` command above. **Ans: False**

True False

(f) (1 pt) In order to profile `bsgsim` with `gprof`, we would have to add a special flag to each `Makefile` command above. **Ans: True**

True False

(g) (1 pt) Suppose `Battlestar.o, Viper.o`, and `Pilot.o` were each placed in static libraries (`libB.a`, `libV.a`, and `libP.a`). If we link them to `bsgsim` with the flags `-lP -lB –lV -lB -lV`, (in that order) we are guaranteed success.

Ans: False

True False

Name: _____

5. (**9 Points**) The project in the previous problem has been imported into a Subversion repository. Bob and Kate both have working copies of this project. All files in the repository and in both of their working copies are at revision 1. Both Bob and Kate begin to edit the file `Pilot.cpp`. After each command executed by Bob or Kate below, indicate the revision of `Pilot.cpp` that appears in the repository, Bob's working copy, and Kate's working copy. If the revision has no number, write the letter "e" (for "edited") in the blank. You may assume that no other commands are executed.

	Repository	Bob's	Kate's
(a) (2 pts) Bob makes edits and executes this command: `svn commit Pilot.cpp -m "Bob"`	_____	_____	_____
Ans:	**2**	**2**	**1**
(b) (2 pts) Kate makes edits and executes this command: `svn revert Pilot.cpp`	_____	_____	_____
Ans:	**2**	**2**	**1**
(c) (2 pts) Kate makes edits and executes this command: `svn commit Pilot.cpp -m "Kate"`	_____	_____	_____
Ans:	**2**	**2**	**e**
(d) (2 pts) Kate executes this command: `svn update Pilot.cpp`	_____	_____	_____
Ans:	**2**	**2**	**e**

(e) (1 pt) Assuming Kate makes no edits to other files in the project, what is the revision of these other files after the above commands are executed?

Answer: The question is ambiguous. In Kate's working copy, subversion will think that the revision number is 1, because she did not update the entire directory. However, these files will be identical to files in the repository, which have the latest revision number. All valid answers were accepted.

Explanation for parts a-d
(a) The file commits successfully, updating the repository and setting Bob's working copy to revision 2.
(b) Kate makes changes, which would have meant her version would change to "e", but she reverts them, which resets her version back to what it was, revision 1.
(c) Kate editer her file (changing it to "e") but when she tries to commit her change, but she is notified of the conflict.
(d) Kate updates her change, and the system attempts to merge her changes with Bob's. Her version is still "e", and she needs to attempt another commit.

Name: _____

6. (**20 Points**) Consider the following code snippets contained in the files indicated. (Feel free to rip out this page.)

```
-- Viper.h -----------------------------

class Viper {
  Battlestar *base;
  int id;
public:
  Viper(int i, Battlestar *b=NULL);
  void Land(Battlestar &b);
  void Launch();
  const Battlestar * GetBattlestar() const;
};

-- Battlestar.h ------------------------

class Battlestar {
  string name;
  vector<Viper*> inside;
public:
  Battlestar(string n, int nVipers=10);
  Viper *GrantLaunchClearance(Viper *v);
  void GrantLandingClearance(Viper *v);
  bool ViperInside(Viper *v) const;
  const string &GetName() const;
};

-- Battlestar.cpp ----------------------

// Lands the Viper v on this Battlestar.
// Precondition: v Cannot be in this Battlestar's list
// Precondition: v's Battlestar must be NULL
void Battlestar::GrantLandingClearance(Viper *v) {
  assert(!ViperInside(v) && v->GetBattlestar() == NULL);
  v->Land(*this);
  inside.push_back(v);
}

// Returns true if the Viper is inside this Battlestar.
// Otherwise, returns false.
bool Battlestar::ViperInside(Viper *v) {
  for (unsigned int i=0; i<inside.size(); i++) {
      if (inside[i] == v)
      return true;
  }
  return false;
}
```

Name: _____

Continue to the next page.

Name: _____

Problem 6, continued:

(a) (6 pts) Given the code above and the code sequence below, indicate whether or not the commented lines below would produce errors when compiled. You may assume that all the necessary headers have been included. (Indicate your choice by writing "Yes" or "No" after each "`// Error?`" comment.)

```
Viper v1 = 1;                                    // Error? No
Viper v2(2);
Battlestar g("Galactica");

Battlestar *b1 = v1.GetBattlestar();             // Error? Yes

const Battlestar *b2 = v1.GetBattlestar();       // Error?  No

cout << v1.GetBattlestar()->GetName();           // Error? No

v1.GetBattlestar()->GrantLaunchClearance(&v1);//Error? Yes

v2.Land(*b2);                                    // Error? Yes
```

(b) (3 pts) Which of the following make good "black box" tests for the method `Battlestar::GrantLandingClearance`? Circle all that apply.

1. A code sequence that checks the effects of the method against the specification.

2. A code sequence that exercises every function call inside the method.

3. A code sequence that violates the method's preconditions.

Solution: Only the first makes a good black box test. The second is a white box test, and the third is not a good test at all, because the result is unpredictable. The previous quarter's final had a question that conflicts with our solution, however, and we therefore accepted 3 as a test.

Name: _____

(c) (5 pts) Suppose that you compile and run the project `bsgsim` with no arguments, after which you discover that `Battlestar::GrantLandingClearance` is failing an assertion. How would you use `gdb` to determine which precondition was violated and which caller violated it? You must give either specific commands or clear descriptions of each step.

There are numerous answers. Here is one:
- Run the program in gdb, wait for assert to trigger
- Use "up" to get to the right stack frame
- Use "p v->base" to check one precondition
- Use "where" to find the caller that violated

Points were removed for out-of-order instructions or missing information. Two points were lost for solutions that require breaking multiple times.

(d) (3 pts) After getting `bsgsim` to work, you start to think that it is running too slowly. You start to profile it using `gprof`. You suspect that the method `Battlestar::ViperInside` is too slow and that you need to implement a faster algorithm. You profile it, check the flat profile, and notice that this method is called 100 times more than any other. Does this give you a clear sense of whether or not you should rewrite this method? Explain your answer.

Solution: No, just because the function is called often does not mean the program is spending lots of time there.

(e) (3 pts) Continuing your investigation of `Battlestar::ViperInside`, you notice that whenever you run the program, about 80% of the time is spent in this method, and it is called only from `Battlestar::GrantLandingClearance`. Does this give you a clear sense of whether or not you should rewrite the method? Explain your answer.

Ambiguous Question:
If "the method" == GrantLandingClearance, then this information gives us no good reason to rewrite it, assuming we wish to keep the assertion. It could call into question our decision to keep the assertion, though.

If "the method" == ViperInside, use this solution: It depends on whether or not the assert will be called in the final version. If it will be, then we have enough information to say that we should optimize it. However, if you mention that you intend to compile with NDEBUG defined (which removes assertions), then this check would not be called in the final version. In this case, you have enough information to say that we should not optimize it.

Name: _____