

# CS 211 Winter 2005

## Final Exam

Instructor: Brian Dennis, Assistant Professor  
Teaching Assistant: Joe Paris

March 17, 2005

**Your Name:**

There are 7 problems on this exam. 4 are worth 10 points apiece and 3 are worth 20 points apiece for a total of 100 points. They are not all of equal difficulty. Please read all of them before starting and attempt the least difficult first.

You have 120 minutes to complete the exam. **When time is up, please stop writing and turn in your quiz. If you have to be asked to stop you will receive no credit for the question you are working on.**

Problem 1	
Problem 2	
Problem 3	
Problem 4	
Problem 5	
Problem 6	
Problem 7	
Total	

**Problem 1**

**10 Points total.**

**5 Points** Briefly explain why overloaded output and input operators are often declared as friends of a class.

**5 Points.** Briefly explain the difference between “call by value” and “call by reference”.

## Problem 2

```
-----  
// classes.h  
class Base {  
public:  
    Base();  
    virtual void howdy();  
    void doody();  
private:  
    // ...  
};  
  
class Derived : public Base {  
public:  
    Derived();  
    virtual void howdy();  
    void doody();  
private:  
    //...  
};  
-----  
  
// base.cpp  
#include <iostream>  
#include "classes.h"  
using namespace std;  
// ...  
void Base::howdy() {  
    cout << "Bacon tastes good. Pork chops taste good." << endl;  
};  
  
void Base::doody() {  
    cout << "You know like Caine in 'KUNG FU'" << endl;  
};  
-----  
  
// derived.cpp  
#include <iostream>  
#include "classes.h"  
using namespace std;  
// ...  
void Derived::howdy() {  
    cout << "Any time of the day is a good time for pie." << endl;  
}  
  
void Derived::doody() {  
    cout << "You guys going to a volleyball game?" << endl;  
}
```

**20 points total**

Assuming the code on the previous page, for each of the numbered lines below indicate what is printed out (2 points) and give a brief explanation (3 points).

```
// -----  
  
int main(int argc, char **argv) {  
    Derived dr;  
    Base& br = dr;  
    Base b = dr;  
    Base* bp = &b;  
    Base* bp2 = &dr;  
  
    br.howdy(); // 1  
    b.howdy(); // 2  
  
    bp->doody(); // 3  
    bp2->doody(); // 4  
    return 0;  
}
```

### Problem 3

```
#ifndef _SET_H_
#define _SET_H_ 1
#include <iostream>
using namespace std;

class BitSet {
public:
    BitSet();

    void add(int d);
    void del(int d);
    bool is_member(int d);
    BitSet unionOfSets(const BitSet& bs);
    BitSet intersectionOfSets(const BitSet& bs);

public:
    static const int ARRAY_SIZE = 256 / 32;
    unsigned int *bits;
};

-----
#include "BitSet.h"

BitSet::BitSet() {
    bits = new unsigned int[ARRAY_SIZE];
};

bool BitSet::is_member(int d) {
    int word = d / 32;
    int offset = d % 32;
    unsigned int mask = 1 << offset;
    int test = bits[word] & mask;
    return (test ? true : false);
}
```

**20 points total**

On the previous page is the class declaration for a `BitSet` class that uses an array of integers as a collection of bit vectors representing a set. We've also included the implementation for the member function `is_member`.

Write 5 member functions (**4 points apiece**) for the `BitSet` class.

- A copy constructor
- An assignment operator
- A destructor
- `add` which takes an integer `d` and adds it to the set
- `del` which takes an integer `d` and deletes it from the set.

More space for problem 3.

## Problem 4

20 points total

```
#ifndef _BASESET_H_
#define _BASESET_H_ 1

template<class T>
class BaseSet {
public:
    virtual bool is_member(T v) = 0;
    virtual void add(T v) = 0;
    virtual void del(T v) = 0;
};
#endif
```

**10 points.** The above class `BaseSet` represents the interface to sets that can hold instances of arbitrary types (although any given set is homogenous e.g. all ints, or all floats, or all strings).

Write the class **declaration** for a concrete, **derived**, **template** class `AnySet` that inherits from `BaseSet` and that we can create instances of. `AnySet` can use composition, or multiple inheritance for the really brave, to get the job done. You'll need at least 4 declared member functions for `AnySet`. This class has no relation to `BitSet` from the previous problem. You are allowed to use classes from the Standard Template Library. If you can't remember a particular template class or member function of that class give a small explanation and we'll try to judge whether you've made a reasonable guess. *Hint: vector is your friend.*

**10 points.** Write **implementations** for the member functions in your class declaration `is_member`, `add`, and `del` from the previous part of the question.

**Problem 5**

**10 Points total.**

**5 Points** Name one type of common pointer error and give an example of the error.

**5 Points** Give a brief example of when you might choose to use inheritance instead of composition in designing a class.

## Problem 6

**10 points** For each of the following statements indicate whether it is true or false (1 point) and give a brief explanation or example (1 point).

A class declaration with a pure virtual member function declaration automatically makes the class an abstract base class.

Space for pointers allocated using `new` are deallocated when the procedure holding the corresponding `new` call returns.

The C++ collection classes, like `vector` and `stack`, are implemented using templates.

Member functions in base classes can't be overridden by derived classes.

`friend` status can only be granted, never taken.

**Problem 7**

**10 points total. 5 Points** Explain what a null terminated string is and sketch one out for the string "WildKats!"

**5 Points.** Discuss why null terminated strings, character arrays, and character pointers all seem interchangeable in C/C++.