CSE 333 Final Exam
June 6th, 2012

Your Name: _____

Student ID: _____

This is a closed book examination. You have **90 minutes** to answer as many questions as possible. The numbers in parentheses at the beginning of each question indicate the number of points given to the question. There are **13 pages** on this exam (check to make sure you have all of them), and there is a total of **100 points** in all. Write all of your answers directly on this paper. Make your answers as concise as possible. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer.

If you need to tear a page out of the midterm to avoid flipping back and forth, do so carefully; don't tear out any pages with your writing on it, and be careful of the staple!

	Points available	**You earned**
Problem 1	24	
Problem 2	24	
Problem 3	24	
Problem 4	24	
Problem 5	4	
Total:	**100**	

Problem 1: Multiple Choice Madness (24 points)

Circle exactly one answer for each of the following questions:

i. It is possible for C code to determine the endian-ness of the underlying CPU.

a) true b) false

ii. In C, a pointer is a variable that contains an address. If you add 2 to a pointer, then:

a) the resulting value is the address plus 2

b) the resulting value depends on what value the pointer points to

c) the resulting value depends on the type of the pointer

d) a segmentation fault is thrown

iii. When you pass a struct as an argument to a C function, then:

a) the struct is passed by value (i.e., a copy of the struct is made, including copying each field in the struct))

b) the struct is passed by reference (i.e., a pointer to the struct is passed)

c) a compiler error is thrown, since you cannot pass structs as arguments

d) what happens depends on the type of fields in the struct

iv. When you pass an array as an argument to a C function, then:

a) the array elements are passed by value (i.e., a copy of the array is made, including copying each element of the array)

b) since arrays are really just pointers, a pointer to the first element of the array is passed and no array elements are copied

c) a compiler error is thrown, since you cannot pass arrays as arguments

d) what happens depends on the type of the array

v. The purpose of a header guard is to:

a) prevent more than one .c file from including a particular .h file

b) prevent the header file from being included indirectly, as a side-effect of including some other .h file that includes it

c) document the contents and purpose of the header file

d) prevent the header file from being included twice, directly or indirectly

vi. A C++ reference:

a) serves as an alternative name for an object or variable (i.e., is an alias)

b) serves as a pointer to an object or variable

c) cannot be used as a parameter of a function

d) cannot be passed as an argument to a function

vii. What does "const" in the following code imply?

```
void foo (const int *x) {   …   }
```

a) the value of the pointer "x" cannot be changed inside the function foo

b) the function foo cannot have any side-effects

c) nothing; const in this case has no effect

d) the value that the pointer "x" points to cannot be changed inside the function foo

viii. What does "const" in the following code imply?

```
void Foo::bar (int *x) {   …   } const;
```

a) the method bar( ) cannot mutate any of its parameters

b) the method bar( ) cannot have any side-effects at all

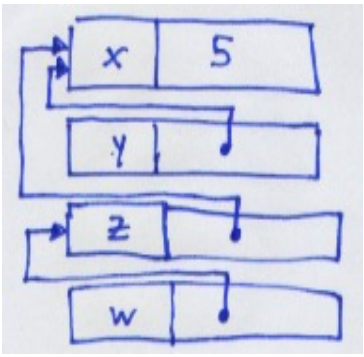
c) the method bar( ) cannot mutate any of Foo's state

d) the method bar( ) can only invoke const-y functions and methods

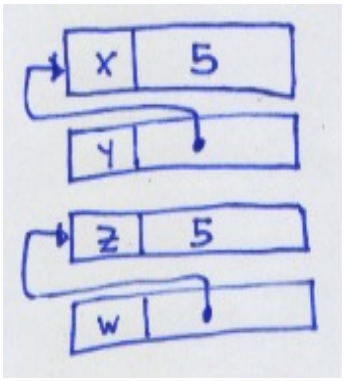
ix. Which of the following box and arrow diagrams correctly represents the following code?

```
int x = 5;
int *y = &x;
int &z = x;
int *w = &z;
```

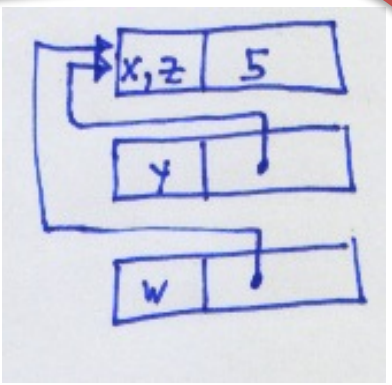
a)



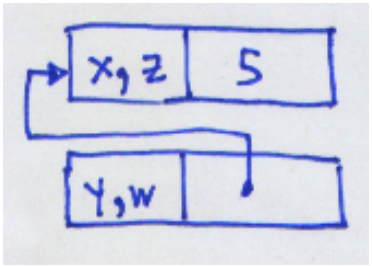
b)



c)



d



x. The destructor of an object that is heap-allocated:

a) is invoked when the function in which it is allocated returns

b) is never invoked

c) must be invoked manually

d) is invoked when somebody uses "delete" to deallocate the object

xi. A vtable:

a) exists for each class, and contains a function pointer for each method in the class

b) exists for each class, and contains a function pointer for each virtual method in the class

c) exists for each object instance, and contains a function pointer for each method in the object's class

d) exists for each object instance, and contains a function pointer for each virtual method in the object's class

xii. Slicing occurs when:

a) the value of a derived class is assigned to an instance of a base class

b) a pointer to a derived class is cast to, and assigned to, a pointer to a base class

c) an N-element array is assigned to an M-element array, where M<N

d) an element is deleted from a std::vector

Problem 2: Virtual and static dispatching (24 points)

On the next page, write the output of the following C++ code:

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

class B {
 public:
  B() { cout << "B::cons" << endl; }

  void p() { cout << "B::p" << endl; }

  virtual void q() { cout << "B::q" << endl; }

  void operator=(B &rhs) { cout << "B::=" << endl; }
};

class Der : public B {
 public:
  Der() { cout << "Der::cons" << endl; }

  void p() { cout << "Der::p" << endl; }

  void q() { cout << "Der::q" << endl; }

  void operator=(Der &rhs) { cout << "Der::=" << endl; }
};

int main(int argc, char **argv) {
  B base1, base2, *baseptr;
  Der der1;

  base2 = base1;
  base2.p();
  base2.q();

  baseptr = (B *) &der1;
  baseptr->p();
  baseptr->q();

  der1.p();
  der1.q();
  base1 = der1;
}
```

Continued on the next page

(Problem 2 continued …)

Write the output here:

```
B::cons
B::cons
B::cons
Der::cons
B::=
B::p
B::q
B::p
Der::q
Der::p
Der::q
B::=
```

Problem 3: Short answers. (24 points)

For each of the following , write a short **(NO MORE THAN TWO SENTENCE!)** answer.

a. What is the purpose of C++'s "virtual" keyword?

The "virtual" keyword tells C++ to use dynamic binding when invoking a function. When invoking a function through a pointer to an object, if the pointed-to object declares the function as virtual, or an ancestor has declared it as virtual, then the pointed-to object's function will be invoked, regardless of the type of the point.

b. Describe one way in which a C++ pointer and a C++ reference behave differently, besides the syntax for using them.

There are several differences, including:

- **a pointer is just a variable containing an address; you can assign and reassign any value to that variable. A reference is an alias for an existing variable; once bound, it cannot be re-bound.**
- **a pointer can be "dangling" if the object/value to which it is pointing is deallocated. It is much harder (but not impossible) to create a dangling reference.**
- **using references, it is easy to write a generic function (i.e., using templates) that works on variables passed by value or by reference, as the syntax for using either is the same. It is much harder to do this with pointers.**

c. What is the difference between C++'s copy constructor and assignment operator?

The assignment operator is invoked when an existing, already initialized object is the LHS of the "=" operator; its job is to assign to the existing objects values drawn from the RHS of the "=" operator.

A copy constructor is invoked to construct a new object, passing an existing object as an argument to the constructor.

d. What is the difference between a thread and a process?

Both a thread and a process are a unit of scheduling and execution, but each process has its own address space, whereas threads share an address space. As well, each process has its own process ID, file descriptor table, and other OS state, whereas threads share much of this state.

e. Why is it usually a good idea to use smart pointers with STL containers?

STL containers store copies of items, and as the containers are accessed and mutated, multiple additional copies can be made. To avoid this, while still allowing the containers to control when items are deallocated, you can store a shared_ptr of an object instead of the object itself.

f. Describe a scenario where makes sense to use a double pointer (i.e., a pointer to a pointer).

There are several scenarios, but two obvious ones are: (a) when you want to return a pointer as an output parameter, such as when the callee dynamically allocates memory or an object and returns a pointer to it; or, (b) when you are manipulating an array of arrays.

Problem 4: Write code. (24 points)

Assume that somebody gives you a library containing this routine:

```
// Reads the next word from a file, and returns a copy of that word
// via the output parameter "next_word".   Returns "true" on
// success, and "false" if there is not a next word, i.e., if we hit
// the end of the file.

bool GetNextWord(std::string *next_word);
```

Make use of this routine to write a C++ program that uses the STL "map" container to keep track of how often each word is used in the file, and then prints each word and the number of times it was spotted.

Remember that:

- a map is kind of like a hash table; it maps a key to a value
- you can access a value of the map given a key using the "[ ]" operator
- if you use "[ ]" to try to access a key that doesn't yet exist, one is created in the map using the default constructor of the value
- if you can't remember precisely how to use a map, invent the syntax you need even if it might not be correct, but note when you're doing this

We've started the program for you, and we have given you two blank pages to work with.

```
#include <iostream>
#include <map>
#include <string>


#include "GetNextword.h"    // defines GetNextWord( )

using namespace std;
```

Continued on the next page

(Problem 4 continued…)

```
int main(int argc, char **argv) {
  // This map tracks the number of times a word is spotted.
  map<string,int> wordcount;

  // This will hold the next word fetched from the document.
  string nextword;

  // Loop fetching the next word, until there is none.
  while(GetNextWord(&nextword) == true) {
    // Increment the count for the next word.  Note that
    // if nextword has not been spotted before, accessing
    // wordcount[nextword] will initialize the value to 0.
    wordcount[nextword] = wordcount[nextword] + 1;
  }

  // Iterate through the word count map printing the results.
  // The problem didn't specify a sort order, so we'll print
  // in whatever order the map iterator returns elements to us.
  map<string,int>::iterator it;
  for (it = wordcount.begin(); it != wordcount.end(); it++) {
    cout << it->first << ": " << it->second << endl;
  }

  // Return "EXIT_SUCCESS", which is 0.
  return 0;
}
```

Continued on the next page

(Problem 4 continued…)

Problem 5: Free points (4 points)

You've made it to the end of the exam!

Write a snippet of C++ code that prints out how many free points you'd like us to give you, constrained to the range [0, 4].

ANYTHING GOES HERE!