COM S 213 PRELIM EXAMINATION #2
April 25, 2002

Name:

Student ID:

Please answer all questions in the space(s) provided. Each question is worth 4 points. You may leave when you are finished with the exam.

Question 1.

In C++, an *interface* is more of a concept rather than a language construct. It involves defining a class with only member functions. In what manner are those member functions defined?

Question 2.

Early on in the year, we learned that standard inheritance involved using the `public` keyword followed by the name of the base class. What other two keywords can appear *instead* of `public`?

Question 3.

Simple recursion involves a function which does what?

Question 4.

Given the following function definition:

```
class Example
{
public:
  void setCounter(int argCounter=kInitialValue);

  enum CounterValues {
    kInitialValue=0, kLastvalue
  };

private:
  int counter;
};

void main()
{
  Example foo;
}
```

How can I call the member function `setCounter` from `main()` to set `foo`'s counter member variable to the constant `kLastValue`? (You may not hard code the constant "1" anywhere in your answer).

Question 5.

Given the following code:

```
class BoringClass
{
public:
  BoringClass() {}
  BoringClass(int initialX) { x = initialX;}
  int getX() { return x; }
private:
  static int x;
};

void main()
{
  BoringClass c1(5);
  BoringClass c2(8);
  cout << "c1.x is: " << c1.getX() << endl;
  cout << "c2.x is: " << c2.getX() << endl;
}
```

What line of code *must* be added in order to make the linker happy?

Question 6.

When you add the line of code necessary to Question 5 to make the linker happy, what will the output be when this program is executed?

Question 7.

Write a template function called `isGreaterThan()` which takes two arguments (`arg1` and `arg2`). Use a template argument to declare the type of those two arguments. Then, `isGreaterThan()` will return a `bool` and should return true if `arg1` is greater than `arg2`. Assume that `operator>` is defined for whatever type the arguments are declared as via the template argument.. Utilize specialization to override the template function when dealing with C-style strings (Remember, when comparing C-style strings, `strcmp(str1,str2)` returns a positive integer if `str1` is "greater than" `str2`)

Question 8.

Show how you would call the template function in Question 7 to compare the two strings below in such a way that the resulting output makes sense:

```
void main()
{
  string str1 = "Hello";
  string str2 = "World";

  if (_____________________________________________________)
    cout << "str1 is LESS THAN OR EQUAL TO str2" << endl;
  else
    cout << "str1 is GREATER THAN str2" << endl;
}
```

Question 9.

Given the following code:

```
class X
{
public:
  virtual void sayHello()
  {
    cout << "Hello from X" << endl;
  }
};

class Y: public X
{
public:
  void sayHello()
  {
    cout << "Hello from Y" << endl;
  }
};

template<class someType>
void sayHello(someType *theType)
{
  theType->sayHello();
}

int main()
{
  X myX;
  Y myY;

  sayHello<X>(&myY);
  return 0;
}
```

What gets printed out?

Question 10.

Given the following function definitions:

```
template<class T>
void aFunction(T arg)
{
  cout << "We're in the template function" << endl;
}
```

```
void aFunction(char *foo)
{
  cout << "We're in the overloaded function" << endl;
}

void main()
{
  char *myStr = "Hello";
  aFunction(myStr);
}
```

What gets printed out?

Question 11.

Given the following simple class:

```
class MyArray
{
public:
  MyArray();
  ~MyArray();
  int &operator[](int k);
private:
  int arrayStore[66];  // Arbitrary store
};
```

Rewrite the definition (do not define the member function and constructors, just declare them) as a template class which takes the type being stored *and* the size of the static array member variable as template arguments:

Question 12.

Using the definition you created in Question 11 as the array "type", declare a variable named `stringArray` which is an array of type `string` that has a maximum size of 50.

Question 13.

Many C++ compilers allow you to disable support for exceptions at compile time. This means that, when disabled, exceptions cannot be thrown and any attempt to compile code that involves exceptions will generate compile time errors. Given such a scenario, write code to check for failed memory allocation and print out "No Memory Available" if the following memory allocation fails:

```
char *ptr;
ptr = new char [10000000000000000000000000000];
```

Question 14.

Define a class that would be acceptable for representing a "bad index" exception. You should minimally store the offending index and a text message.

Question 15.

Implement the member function `operator[]` from the `MyArray` class you defined in your answer to question 11. Throw an exception using the class defined in question 14 if you detect a bad index.

Question 16.

Given the following code:

```
try
{
  executeSomeFunction();
}
catch(exception e)
{
  throw;
}
```

What (specifically) happens when we encounter the `throw` keyword?

Question 17.

Given the following code:

```
MyArray<int,10> gArray;
gArray[99] = 5;
```

How would I re-write this code to catch the exception thrown when an out of bounds index is detected in the second line?:

Question 18.

What error condition is associated with the `bad_alloc` exception?

Question 19.

Given the following vector declaration:

```
void main()
{
  vector<int> myVector;

  myVector.push_back(5);
  myVector.push_back(10);
  myVector.push_back(15);
  myVector.push_back(20);
}
```

Write the declaration for an `iterator` variable named `p` which can be used to "point" at any of the items in this vector:

Question 20.

Show how you would assign a value to the iterator declared in question 19 such that it would "point" at the first item in the vector (Hint: It should "point" at the beginning of the vector)

Question 21.

Given the following code:

```
typedef map<string,string>::value_type IDRecord;
void main()
{
  map<string,string> ids;

  IDRecord rec1("one","Value1");
  IDRecord rec1("two","Value2");
  IDRecord rec1("three","Value3");
  ids.insert(rec1);
  ids.insert(rec2);
  ids.insert(rec3);
}
```

Write code to print out the value associated with the key `two` in the map named `ids`. (There is more than one answer, any correct answer will be accepted).

Question 22.

Given the following code:

```
int x = CS213::someVar;
```

If `CS213` is **not** a class name, what is it?

Question 23.

What is the significance of the following line of code? (What does it enable us to do?)

```
using namespace std;
```

Question 24.

Why would you ever want to overload the `new` and `delete` operator for a given class?

Question 25.

Create three simple classes named `C1`, `C2` and `C3`. They can contain anything you want them to, but `C3` *must* be derived from both `C1` *and* `C2` via multiple inheritance.