# COM S 213 PRELIM EXAMINATION #2
# April 26, 2001

**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.**

Define an *interface* named `Printable` which has two member functions: `print()` and `printSetup()`. Both member functions take *no* arguments and return a `bool`.

**Question 2.**

Suppose I have a class named `Base`. Show how a can derive a class named `Derived` *privately* from `Base` (you don't need to define any member variables or functions in `Derived`, just show the syntax of private inheritance)

**Question 3.**

If , during the course of compiling, the compiler encounters an `#ifdef` in your code, what other preprocessing command will it expect to find before the compile is finished?

**Question 4.**

Given the following class definition:

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

  enum CounterValues {
    kInitialValue=0,  kLastValue
  };
private:
  int counter;
};

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

How can I set `foo`'s counter value to `kLastValue` from the `main` function (You may not hard code the constant "1" in the call to `setCounter`)

```
foo.setCounter(_____________________________________);
```

**Question 5.**

Given the following code:

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

int BoringClass::x = 0;

void main()
{
  BoringClass c1(5);
  BoringClass::x = 9;
  BoringClass c2(8);
  cout << "c1.x is: " << c1.getX() << endl;
  cout << "c2.x is: " << c2.getX() << endl;
}
```

What gets printed out?

```
c1.x is: _____________________

c2.x is: _____________________
```

**Question 6.**

Given the following code:

```
class Y
{
public:
  void func1()
  { cout << "a,b = " << a << "," << b << endl; }

private:
  int a,b;
};

class X : private Y
{
public:
  void func2()
  { cout << "h,k = " << h << "," << k << endl; }
private:
  int h,k;
};

void main()
{
  X x;
  x.func1();
  x.func2();
}
```

What declaration should appear in the public section of class `X` in order to to avoid any compiler errors when compiling?

**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.

**Question 8.**

Show how you would call the template function written in Question 7 to compare two instances of `MyString` as defined below:

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

  // Compare two elements below

  if (_____________________________________________________________)
      cout << "str1 is GREATER than str2" << endl;
  else
      cout << "str1 is LESS THAN OR EQUAL TO str2" << endl;

}
```

**Question 9.**

Given the following code:

```
class X
{
public:
  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);
}
```

What, if anything, would I need to do in the base class `X` to cause this code to print out "Hello from Y" (besides changing the string that reads "Hello from X" to "Hello from Y" is `X::sayHello()`)

**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;
}

int main()
{
  string myStr = "Hello";
  aFunction(myStr);
}
```

Assuming `MyString` does not have any typecast overloads defined, what gets printed out?

**Question 11.**

Given the following simple integer array class definition:

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

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", how would you declare a variable q which is a list of `MyArray`s of type `float` using the STL `list` as the list type? The size of the `MyArray` of floats in this question is 8. (HINT: Remember talking about template classes as type arguments to other template classes?)

**Question 13.**

In C programming and the early days of C++ programming, we would check to see if dynamic memory allocations failed by seeing if the memory allocation function returned NULL. Given the following code:

```
char *c;
c = new char[10000000000000000];
if (c == NULL)
  cout << "Could not allocate memory" << endl;
else
  // Code continues here
```

Why won't this approach work in today's more modern C++ compilers and run time environments?

**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.**

Complete the following catch block such that the exception that is caught is thrown again:

```
void MoveRobot(Position *positions,int numPos)
{
  Position origPos = getCurrentPosition();
  try {
    for (int i=0; i<numPos; i++)
      MoveRobot(positions[i]);
  } catch(BadPositionException bpe) {
      MoveRobot(origPos);

      ____________________________________________

  }
  MoveRobot(origPos);
}
```

**Question 17.**

Assuming I have a global `MyArray` defined as follows:

```
MyArray<MyString,10> gArray;
```

Write a function called `showMeTheElement()` which takes no arguments, has a void return value, prompts the user for an index and then displays the element from `gArray` at that index. This function should be prepared to deal with the exception you defined in question 14 (and defined rules for throwing in question 15).

**Question 18.**

What is the data type of the exception you should be checking for when trying to detect memory allocation failures? (Hint: it isn't `exception`, but it is derived from `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 rec2("two",  "Value2");
  IDRecord rec3("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:

```
class B
{
public:
  void f() { cout << "She loves me!!!!" << endl; }
};

class D1 : virtual public B
{
public:
  void g() { f(); }
};

class D2 : virtual public B
{
public:
  void f() { cout << "She loves me not" << endl; }
}
class DD: public D1, public D2 { };

int main()
{
  DD me;
  me.g();
  return 0;
}
```

What gets printed out?

**Question 23.**

Given the following namespace declaration:

```
namespace CS213
{
  int x = 5;
};
```

How do I access the variable x without the use of any `using` statements?

---

**Question 24.**

Given the following code:

```
template <class T=MyString>
class SomeClass
{
…
}
```

What is the significance of the `=MyString` that appears after the `class T` in the template argument list?

**Question 25.**

If you would like to write specialized memory allocation routines for a class that you are implementing (and not a predefined class, like the STL containers), what operators must you overload in that class?