1. **True/False.** Circle the appropriate choice on this sheet.

(a) T F One of the supposed benefits of object oriented programming is code reuse.

(b) T F A C++ variable of type T* can have value 0.

(c) T F A C++ variable of type T& can have value 0.

(d) T F The accept method for the visitor design pattern can be implemented by the root of the class hierarchy.

(e) T F If a field of a Java class is final, the referenced object cannot be modified.

(f) T F If a Java class’ only constructor is declared private, no instances of the class can be created.

(g) T F The erasure property of Java generics means that no run-time type information is needed for any Java types.

(h) T F In the subset interpretation of subtyping, a child class defines a set of objects that is a subset of the set defined by the parent class.

(i) T F Scala combines the functional and object oriented language paradigms.

(j) T F In C++, all classes derive, directly or indirectly, from the Object class.

2. (a) In the context of subtyping, what do the terms *covariant*, *contravariant*, and *invariant* mean?

(b) Assume you have a language that allows subtyping among function types. Further, suppose the language is type safe (i.e. never allow access to a field or method of an object that doesn’t actually exist). For (A → B) to be a subtype of (C → D), what must the relationships among the types A, B, C and D be?

(c) Give a very simple programming example, in whatever syntax you like, that illustrates why violating the relationships among the types that you stated in your previous answer (if the language were to allow such violations) could lead to an unsafe access to an object.

(d) Suppose you have a language that does not allow subtyping among function types, but, does allow subtyping among instances of a generic class. In order for the language to be type safe, such that, given a generic class C[T] (where T is the formal type parameter) and assuming B is a subtype of A, C[B] is a subtype of C[A], what must be true about generic class C?

(e) Give a very simple programming example that illustrates why violating the property of generic classes (such as C, above) that you stated in your previous answer could lead to an unsafe access to an object.

3. Consider the following code in C++.

```cpp
class A {
    int x;
public:
    void f();
};
```
virtual void g();
protected:
  int y;
};

class B {
  int z;
public:
  virtual void h();
protected:
  int w;
};

class C: public A {
public:
  void f();
  void g();
protected:
  int q;
};

class D: public A, public B {
public:
  void h();
protected:
  int r;
};

main()
{
  A *p1 = new A();
  C *p2 = new C();
  D *p3 = new D();
}

Draw a picture showing the layout in memory of the objects that p1, p2 and p3 point to. Be sure to draw, in detail, the vtable(s) that each object points to as well. Label all fields of the objects and vtables.

4. (a) In Java, all objects have reference semantics. Is that true in C++? Explain.
(b) Write a minimal C++ class and main function that illustrates all possibilities for reference or value semantics for objects.
(c) In Java, all arrays also have reference semantics. Is that true in C++? Explain.
(d) Write a C++ class that contains at least one array and that illustrates all possibilities for reference or value semantics for arrays.
5. (a) What is the difference between Java’s “final” and C++’s “const”?
   (b) What does it take to ensure “const-correctness”?
   (c) Write a C++ class that illustrates your answer to the previous question.

6. (a) What is the key idea behind the visitor design pattern?

7. (a) At what time are overloaded methods (as opposed to overridden) resolved?
   (b) At what time are overridden (that is, virtual) methods resolved?
   (c) Write one or more Java classes that have overloaded, overridden, as well as overloaded and overridden methods. Annotate each method with a comment identifying whether the method is overloaded, overridden, or both.

8. (a) Why do arithmetic operations on integral types in Java always result in “int” or “long” results?
   (b) How does that impact method overload resolution?
   (c) Write a Java class that illustrates your answers to the previous two questions.

9. (a) When are C++ copy constructors, assignment operators, and destructors, respectively, invoked?
   (b) If a class defines one of these (copy constructor, assignment operator, destructor), it should define all of them (the so-called “law of the big three”). Why?
   (c) class myint {
       int* ptr;
   public:
       myint(int val) : ptr(new int(val)) {}
       int get() {
           return *ptr;
       }
       void set(int val) {
           *ptr = val;
       }
   }
   Write the copy constructor, assignment operator, and destructor for class myint, so that the class has value semantics.

10. Why is the counter of our reference-counting smart pointer allocated on the heap and not inside the smart pointer?

11. (a) Explain in detail the difference, if any, between the following two versions of a Java generic class?
    class C<T> {
        T x;
        void foo(T y) { ... }
    }
    and
class C<T> {
    T x;
    <T> void foo(T y) { ... }
}

(b) What could be written in the body of foo(), replacing the “...” that would cause the Java compiler to accept the first version of C but reject the second version of C?

(c) Given the following Java interface,

```java
interface Addable<T> {
    public T add(T x);
}
```

write a generic static method sum() that can take any ArrayList containing Addable elements and returns the sum of all the elements in the ArrayList. Note that the ArrayList must contain at least one element.

(d) In the following Java method (assuming the above Addable interface is defined),

```java
public static <T> void f(ArrayList<Addable<T>> a, T x, T y) {
    if (x.add(y) != x)
        a.add(y);
}
```

fill in the blank areas between each pair of angle brackets so that code compiles without error. Assume that there is no class T defined anywhere else.

12. Given a template class

```cpp
template<typename T>
class Klass { ... };
```

and two instantiations

```cpp
Klass<int> i;
Klass<double> d;
```

how many copies of Klass’ methods are generated by the C++ compiler?

13. (a) In Scala, define a generic trait that allows objects of a class that implements the trait to be compared to each other.

(b) Using Scala’s “case class” facility, define a generic class for implementing binary search trees where, for any class T implementing the trait you defined in your previous answer, each node of the tree type resulting from instantiating your generic class contains a value of type T.

(c) Add an insert method to your generic binary search tree class for inserting a new value (of type T) into a binary search tree. Be sure to use Scala’s pattern matching facility.

14. Unlike Java, C++, and Scala, JavaScript is interpreted. What is the second major difference from the former three languages?

15. In JavaScript, what is the difference between the “prototype” property on one hand and the “__proto__” property (most browsers) or “Object.getPrototypeOf()” (ECMAScript standard) on the other hand?