Jake Hill
CSE4252
Sangeeta Kumari
Homework 2
Due: 11-6-17 11:59pm

1) What are the different types of inheritance? Give the syntax for each in C++.

Public :

```
class foo : public bar {...}
```

Protected:

```
Class foo : protected bar {...}
```

Private

```
class foo : bar {...} //defaults to private if access control is not defined
```

2) With the discussed example in class (Person as base and Student derived from it), discuss upcasting and downcasting.

Let Person be a base class of Student, where Student is a derived class of Person.

Let both classes share a common method “void show()”, Person has a member variable name, and Student has a member variable major.

Upcasting: This allows a pointer to a base class to be implicitly cast to one of its derived classes,

```
Student s;
Person *pPerson = &s;
```

In the above example, pPerson is allowed to be assigned, without type casting, to the address of a Student object since a Student has the same member functions of Person, plus its own member function(s) defined in its class definition. This however, alludes itself to another trait of this operation called object slicing which I will discuss in question 3.

Downcasting: This allows a pointer to a derived class to be explicitly cast to its base class.

```
Person p;
Student *pStudent = (Student *) &p; // explicit cast is required
```

This operation is dangerous however, since, in our example, you would be allowing a pointer to have access to member functions that the object in which it is pointing to does not actually possess.

I.e. if you called “pStudent -> major; “ this would result in a runtime error since p does not have a major member variable thus, you are accessing memory outside of referenced object’s memory address space.

3) With the knowledge of previous question, explain Object Slicing with an example.

Object Slicing is a byproduct of an upcasting operation such that since you are assigning a reference for a derived class to a base class pointer you are striping the access to unique member functions and variables of that derived class object’s memory space.

For example:

```
Student s;
Person *pPerson = &s;
Cout << s.major; // compiles just fine
Cout << pPerson.major // error class Person has no member named ‘major’
```

4) Can we have virtual constructors? If no, then why not?

No we cannot. Virtual functions are used run common member functions between base and derived classes without knowing the exact type of an object but simply the interface. When creating an object we must know exactly what we are trying to create so that we can allocate the proper memory space for it on the stack or heap. Thus, having a virtual constructor contradicts the requirements of a constructor.

5) What is the difference between overriding and overloading? Give code snippets.

Overriding: To redefine the definition of an existing function or operator

Ex:

```
class foo{
    public:
        void print(){
            cout << "I am foo";
        }
}

class bar : public foo {
    public:
        // Here we overload the function print()
        // originally defined in foo's parent class
        void print(){
            cout << "I am bar";
        }
}
```

Overloading: To define a function more than once but with sets of formal parameters which are different in the number of parameters and the data types which they ask for.

Ex:

```
class foo{
    public:
        void print(){
            cout << "I am foo";
        }
        void print(int bar){
            ...
        }
        void print(double d_bar){
            ...
        }
}
```