1. Given the following class hierarchy, which inherited members can be accessed without qualification from within the VMI class? Which requires qualification? Explain your reasoning.

```
struct Base {
        void bar(int); // public by default
protected:
        int ival;
};
struct Derived1 : virtual public Base {
        void bar(char); // public by default
        void foo(char);
protected:
        char cval;
};
struct Derived2 : virtual public Base {
        void foo(int); // public by default
protected:
        int ival;
        char cval;
};
class VMI : public Derived1, public Derived2 { };
```

ANSWER:

Without Qualification:

ival :- VMI -> Derived2 :: ival-> Base :: ival. Direct path hence precedence.

bar:- VMI-> Derived1 :: bar-> Base :: bar. Direct path hence precedence.

Require Qualification:

cval:- VMI-> Derived2::cval -> Base:: cval

-> Derived1::cval->Base:: cval. Multiple paths hence required.

foo:- VMI-> Derived2::foo -> Base:: foo

-> Derived1::foo->Base:: foo. Multiple paths hence required.

2. Given the following class hierarchy:

```
class Class { ... };
class Base : public Class { ... };
class D1 : virtual public Base { ... };
class D2 : virtual public Base { ... };
class MI : public D1, public D2 { ... };
class Final : public MI, public Class { ... };
```

(a) In what order are constructors and destructors run on a Final object?

(b) A Final object has how many Base parts? How many Class parts?

(c) Which of the following assignments is a compile-time error?

```
Base *pb; Class *pc; MI *pmi; D2 *pd2;
(a) pb = new Class; (b) pc = new Final;
(c) pmi = pb; (d) pd2 = pmi;
```

a) Order of constructors on a final object: Class(); // run by Base default constructor

Base(); // D1 & D2 virtual base class initialized first

D1(); // indirect nonvirtual base class

D2(); // indirect nonvirtual base class

MI(); // first direct nonvirtual base class

Class(); // second direct nonvirtual base class (initialized again)

Final(); // most derived class

Now the destructor will run from last object called to first, Hence

Final(); // most derived class

Class(); // second direct nonvirtual base class (initialized again)

MI(); // first direct nonvirtual base class

D2(); // indirect nonvirtual base class

D1(); // indirect nonvirtual base class

Base(); // D1 & D2 virtual base class initialized first Class(); // run by Base default constructor

b) Final object will have 1 base part and 2 class parts.

c) a) Invalid type conversion: Base object converted to Class object

b) Class is inaccessible directly due to ambiguity. There are multiple paths to Class

c) Invalid type conversion: Base object converted to Class object

d) Valid, pmi not initialized

3. Given the following classes, explain each print function:

```
class base {
        public:
                string name() { return basename; }
                virtual void print(ostream &os) { os << basename; }
        private:
                string basename;
        };
        class derived : public base {
        public:
                void print(ostream &os) { print(os); os << " " << i; }
        private:
                int i;
        };
```

If there is a problem in this code, how would you fix it?

ANSWER:

The base virtual functionprints the value of the base name member. It should be const number because it does not modify any data members.

void base::print(ostream &os) const { os << basename }

The print in derived want to call the print from base class but it’s scope is omitted. Hence it will lead to Segmentation fault: print(os) in derived will call itself indefinitely.

void print(ostream &os) {base::print(os); os <<” “<<i;}

4. Given the classes from the previous problem and the following objects, determine which function is called at run time:

```
        base bobj; base *bp1 = &bobj; base &br1 = bobj;
        derived dobj; base *bp2 = &dobj; base &br2 = dobj;
    (a) bobj.print(); (b) dobj.print(); (c) bp1->name();
    (d) bp2->name(); (e) br1.print(); (f) br2.print();
```

ANSWER:

e) br1.print();

and

(f) br2.print();

are called at run time.