Name: Brijesh Rameshbhai Rohit

Admission number: U19CS009

PRINCIPLE OF PROGRAMMING LANGUAGES

ASSIGNMENT - 05

Q1. 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);
protected:
   int ival;
};
struct Derived1 : virtual public Base
{
   void bar(char);
   void foo(char);
protected:
   char cval;
};
struct Derived2 : virtual public Base
{
   void foo(int);
protected:
   int ival;
   char cval;
};
class VMI : public Derived1, public Derived2
{

};
```

For object of VMI class:Data member ***'ival'*** inherited from class ***'Derived2'*** and function ***'bar(char)'*** of class ***'Derived1'*** would not need and qualification in order to access them.

All the other data members and functions will require qualification to access them:

- ***{Base} ival :***
- ***{Derived1} foo(char):***
- ***{Derived1} cval:***
- ***{Derived2} cval:***

Q2. 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?

Order of constructor call:

1. Constructor of Class.
2. Constructor of Base.
3. Constructor of D1.
4. Constructor of D2.
5. Constructor of MI.
6. Constructor of Class
7. Constructor of Final.

Order of Destructor call:

1. Destructor of Final.
2. Destructor of Class.
3. Destructor of MI.
4. Destructor of D2.
5. Destructor of D1.

6. Destructor of Base.

7. Destructor of Class.

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

1 Base part and 2 Class part.

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

Base *pb; Class *pc; MI *pmi; D2 *pd2;

(a) pb = new Class; **Error: Child class cannot be pointer of parent class.**

(b) pc = new Final; **Allowed.**

(c) pmi = pb; **Error: A class cannot be pointer of predecessor class.**

(d) pd2 = pmi; **Allowed.**

==========

Q3. 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?

=> The call of print function from object of class derived would result in an infinite recursive call and hence there would not be any output and time limit would exceed.

This is happening as the 'print(os);' int the print function of derived is call the print function of derived instead of the print function of base class. To prevent this we just need to replace the statement of 'print(os);' with 'base:: print(os);'

So the new code would look like:

```
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)
   {
      base::print(os);
      os << " " << i;
   }
private:
   int i;
};
```

Q4. 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();

=>

- As we have used virtual function for overriding the print function, thus the functions which would be called at runtime would be 'print()'. Thus, we can eliminate (c) and (d).
- (a) can be eliminated as bobj is object of Base class and the print function here is called statically.
- (b) is of derived class only and no reference or pointer of base is pointed to derived so it will be statically bounded only.
- (e) is reference to derived from base class, and being a reference it will access data in run time only. So (e) will be dynamically bounded. Same goes for (f).

Thus (e) and (f) would be called at run time.