**Multiple Inheritence**

**Writer:** Sayak Haldar

**1. What is meant by multiple inheritance?**

a) Deriving a base class from derived class

b) Deriving a derived class from base class

c) Deriving a derived class from more than one base class

d) None of the mentioned

**Answer: c**

Explanation: Multiple inheritance enables a derived class to inherit members from more than one parent.

**2. Which symbol is used to create multiple inheritance?**

a) Dot

b) Comma

c) Dollar

d) None of the mentioned

**Answer: b**

Explanation: For using multiple inheritance, simply specify each base class (just like in single inheritance), separated by a comma.

**3. Which of the following advantages we lose by using multiple inheritance?**

a) Dynamic binding

b) Polymorphism

c) Both Dynamic binding & Polymorphism

d) None of the mentioned

**Answer: c**

Explanation: The benefit of dynamic binding and polymorphism is that they help making the code easier to extend but by multiple inheritance it makes harder to track.

**Now, what is dynamic binding?**

In dynamic binding the function call matches with the correct function definition at runtime. It is also known as Dynamic Binding or Runtime Binding. Now, it is another feature provided by polymorphism. The base class pointer can be initialised with the derived class object dynamically. Now, when a function called using Base class’ pointer, in compilation time only one thing is checked. Whether the function declaration is present in the base class. (considering that the function could be a pure virtual function). However, it runtime, it is checked whether the derived class overrides the function definition provided by base class (if any) or not. If it is overridden, the derived class definition is called. If it is not overridden base class’s definition is called. (if it is provided in Base class). So, it is runtime when this is decided. That is why it is called dynamic binding/late binding.

**It cannot be done since multiple base class is present. For similar reason, we cannot use Base class pointer to hold the reference of a derived class object or to create a derived class object dynamically**

**4. What is the output of this program?**

#include <iostream>

using namespace std;

class polygon

{

protected:

int width, height;

public:

void set\_values (int a, int b)

{

width = a; height = b;

}

};

class output1

{

public:

void output (int i);

};

void output1::output (int i)

{

cout << i << endl;

}

class rectangle: public polygon, public output1

{

public:

int area ()

{

return (width \* height);

}

};

class triangle: public polygon, public output1

{

public:

int area ()

{

return (width \* height / 2);

}

};

int main ()

{

rectangle rect;

triangle trgl;

rect.set\_values (4, 5);

trgl.set\_values (4, 5);

rect.output (rect.area());

trgl.output (trgl.area());

return 0;

}

a) 20

b) 10

c) 20

10

d) None of the mentioned

**Answer: c**

**5. What is the output of this program?**

#include <iostream>

using namespace std;

class Base

{

public:

virtual void print() const = 0;

};

class DerivedOne : public Base

{

public:

void print() const

{

cout << "DerivedOne\n";

}

};

class DerivedTwo : public Base

{

public:

void print() const

{

cout << "DerivedTwo\n";

}

};

class Multiple : public DerivedOne, public DerivedTwo

{

public:

void print() const

{

DerivedTwo :: print();

}

};

int main()

{

int i;

Multiple both;

DerivedOne one;

DerivedTwo two;

Base \*array[ 3 ];

array[ 0 ] = &both;

array[ 1 ] = &one;

array[ 2 ] = &two;

array[ 0 ] -> print();

return 0;

}

a) DerivedOne

b) DerivedTwo

c) Error

d) None of the mentioned

**Answer: c**

Explanation: In this program, ‘Base’ is an ambiguous base of ‘Multiple’. So it is producing an error.

**We need the feature called virtual base class.**

**6. What is the output of this program?**

#include <iostream>

using namespace std;

class student

{

public:

int rno , m1 , m2 ;

void get()

{

rno = 15, m1 = 10, m2 = 10;

}

};

class sports

{

public:

int sm;

void getsm()

{

sm = 10;

}

};

class statement:public student,public sports

{

int tot,avg;

public:

void display()

{

tot = (m1 + m2 + sm);

avg = tot / 3;

cout << tot;

cout << avg;

}

};

int main()

{

statement obj;

obj.get();

obj.getsm();

obj.display();

}

a) 3100

b) 3010

c) 2010

d) 1010

**Answer: b**

Explanation: In this program, We are calculating the total and average marks of a student by using multiple inheritance.

**7. What is the output of this program?**

#include <iostream>

using namespace std;

struct a

{

int count;

};

struct b

{

int\* value;

};

struct c : public a, public b

{

};

int main()

{

c\* p = new c;

p->value = 0;

cout << "Inherited";

return 0;

}

a) Inherited

b) Error

c) Runtime error

d) None of the mentioned

**Answer: a**

Explanation: In this program, We apply the multiple inheritance to structure.

**8. What is the output of this program?**

#include <iostream>

using namespace std;

class Base1

{

protected:

int SampleDataOne;

public:

Base1()

{

SampleDataOne = 100;

}

~Base1()

{

}

int SampleFunctOne()

{

return SampleDataOne;

}

};

class Base2

{

protected:

int SampleDataTwo;

public:

Base2()

{

SampleDataTwo = 200;

}

~Base2()

{

}

int SampleFunctTwo()

{

return SampleDataTwo;

}

};

class Derived1 : public Base1, public Base2

{

int MyData;

public:

Derived1()

{

MyData = 300;

}

~Derived1()

{

}

int MyFunct()

{

return (MyData + SampleDataOne + SampleDataTwo);

}

};

int main()

{

Base1 SampleObjOne;

Base2 SampleObjTwo;

Derived1 SampleObjThree;

cout << SampleObjThree.Base1 :: SampleFunctOne() << endl;

cout << SampleObjThree.Base2 :: SampleFunctTwo() << endl;

return 0;

}

a) 100

b) 200

c) Both 100 & 200

d) None of the mentioned

**Answer: c**

Explanation: In this program, We are passing the values by using multiple inheritance and printing the derived values.

**9. Which design patterns benefit from the multiple inheritance?**

a) Adapter and observer pattern

b) Code pattern

c) Glue pattern

d) None of the mentioned

**Answer: a**

**10. What are the things that are not inherited from the base class?**

a) Constructor and its destructor

b) Operator=() members

c) Friends

d) All of the mentioned

**Answer: d**