Group Activity 07

(3인 혹은 4인으로 팀을 구성하여 아래의 문제를 푼다. 팀 구성은 매 시간마다 달라져도 된다.)

팀원1: ____________________

팀원2: ____________________

팀원3: ____________________

팀원4: ____________________

1. 다음 프로그램의 출력은? 컴파일 오류나 실행 오류가 나는 경우에는 이유를 간략히 설명하라.

Program	Output
```
class Base {
public:
  Base() {
    cout << "Base constr called" << endl;
  }
};

class Derived: public Base {
public:
  Derived() {
    cout << "Derived constr called" << endl;
  }
};

int main() {
   Derived d;
   return 0;
}
```
```
class Base {
  int arr[10];
};

class Derived: public Base {
  int d;
};

int main() {
  cout << sizeof(derived);
  return 0;
}
```

```
class P {
public:
  void print()  {
    cout << "Inside P";
  }
};

class Q: public P {
public:
  void print() {
    cout << "Inside Q";
  }
};

class R: public Q { };

int main() {
  R r;
  r.print();
  return 0;
}
```

```
class Base {
private:
  int x = 1, y = 2;
};

class Derived: public Base {
public:
  void show(){
    cout << x << " " << y << endl;
  }
};

int main(void) {
    Derived d;
    d.show();
    return 0;
}
```

```
class Base {
protected:
  int x, y;
public:
  Base(int a=1, int b=2): x(a), y(b) { }
};

class Derived: public Base {
public:
  void show(){
    cout << x << " " << y;
  }
};
int main(void) {
  Derived d;
  d.show();
  return 0;
}
```

```cpp
class Base {};

class Derived: public Base {};

int main() {
  Base *bp = new Derived;
  Derived *dp = new Base;
}
```

```cpp
class Base {
public:
  void show() {
    cout << "In Base";
  }
};

class Derived: public Base {
public:
  int x;
  void show() {
    cout << "In Derived";
  }
  Derived() {
    x = 10;
  }
};

int main() {
  Base *bp, b;
  Derived d;
  bp = &d;
  bp->show();
  cout << bp->x;
  return 0;
}
```

```cpp
class Base {
public:
    int fun()  {
        cout << "Base::fun() called";
    }
    int fun(int i)  {
        cout << "Base::fun(int i) called";
    }
};

class Derived: public Base {
public:
    int fun() {
        cout << "Derived::fun() called";
    }
};

int main() {
    Derived d;
    d.fun(5);
    return 0;
}
```

```
class Base {
public:
  void fun()  {
    cout << "Base::fun() called";
  }

  void fun(int i)  {
    cout << "Base::fun(int i) called";
  }
};

class Derived: public Base  {
public:
  void fun()   {
    cout << "Derived::fun() called";
  }
};

int main()  {
  Derived d;
  d.Base::fun(5);
  return 0;
}
```

```
class Base {
public:
  virtual string print() const {
    return "This is Base class";
  }
};

class Derived : public Base {
public:
  virtual string print() const {
    return "This is Derived class";
  }
};

void describe(Base p) {
  cout << p.print() << endl;
}

int main() {
  Base b;
  Derived d;
  describe(b);
  describe(d);
  return 0;
}
```

```
class Base {
public :
  int x, y;
  Base(int a, int b) {
    x = a; y = b;
  }
};

class Derived: public Base  {
public:
  Derived(int p, int q): x(p), y(q) {}
  void print() {
    cout << x << " " << y;
  }
};

int main(void) {
  Derived q(10, 10);
  q.print();
  return 0;
}
```

```
class A {
    float d;
public:
    int a;
    void change(int i) {
        a = i;
    }
    void value_a() {
        cout << a << endl;
    }
};

class B: public A {
    int a = 15;
public:
    void print() {
        cout << a << endl;
    }
};

int main() {
    B b;
    b.change(10);
    b.print();
    b.value_a();
    return 0;
}
```

```cpp
class A {
    double d;
public:
    virtual void func() {
        cout << "In class A\n";
    }
};

class B: public A {
    int a = 15;
public:
    void func() {
        cout << "In class B\n";
    }
};

int main() {
    B b;
    b.func();
    return 0;
}
```

```cpp
class A {
    double d;
public:
    virtual void func() {
        cout << "In class A\n";
    }
};

class B: public A {
    int a = 15;
public:
    void func() {
        cout << "In class B\n";
    }
};

int main() {
    A *a;
    a->func();
    return 0;
}
```

```
class A {
    double d;
public:
    virtual void func() {
        cout << "In class A\n";
    }
};

class B: public A {
    int a = 15;
public:
    void func() {
        cout << "In class B\n";
    }
};

int main() {
    A *a = new A();
    B b;
    a = &b;
    a->func();
    return 0;
}
```

```
class Base {
public:
  ~Base()  {
    cout << " Base destructor" << endl;
  }
};

class Derived: public Base {
public:
  ~Derived()  {
    cout << " Derived destructor" << endl;
  }
};

int main() {
   Derived d;
   return 0;
}
```