OBJECT-ORIENTED PROGRAMMING
HOMEWORK #4
SPRING 2023

DUE: Monday, April 10

(1) Is there a bug(s) in the following code? (If so, explain)

```
class A {
public:
        A() { val = 0; }
        virtual ~A() {}
protected:
        A(int x) { val = x; }
private:
        int val;
};

class B: public A {
public:
        B(): A(0) {}
        A *createA(int x = 0) { return new A(x); }
};
```

(2) What is the output of the following program? Explain why you have this output.

```
#include <iostream>
Using namespace std;

class A {
public:
    void f() { cout << "A::f()\n"; }
    virtual void h(){ cout << "A::h()\n"; }
};

class B: public virtual A {
public:
    void f() { cout << "B::f()\n"; }
};

class C: public B {
public:
    void g() { cout << "C::g()\n"; }
    void h() { cout << "C::h()\n"; }
};

class D: public C, public virtual A {
public:
    void g() { cout << "D::g()\n"; }
    void h() { cout << "D::h()\n"; }

};
```

```
main() {
A *pa = new D; pa->f(); pa->h();
B *pb = new B; pb->f(); pb->h();
C *pc = new C; pc->f(); pc->h();
D *pd = new D; pd->f(); pd->g(); pd->h();

return 0;
}
```

(3) Consider the following hierarchy.

```
struct Base {
        Base() { cout << "A Base is born\n"; }
        ~Base() { cout << "A Base dies\n"; }
        void func() { cout << "Base::func()\n"; }
        virtual void vfunc() { cout << "Base::vfunc()\n"; }
};

struct Der: public Base {
        Der() { cout << "A Der is born\n"; }
        ~Der() { cout << "A Der dies\n"; }
        void func() { cout << "Der::func()\n"; }
        virtual void vfunc() { cout << "Der::vfunc()\n"; }
};
```

Consider the following code.

```
Base *bp;

bp = new Der;
bp->func();
bp->vfunc();
delete bp;
```

(a) What is the output for this code?
(b) Does the output seem correct? If not, how would you fix the problem.

(4) You must solve this problem with a C++ program and with a Python program.

Write an abstract base class called Employee that contains a name and social security number (both are strings) together with appropriate member functions. Also include a pure virtual function (abstract method in Python) printout() that prints out the name and social security number.

Next, derive (from Employee) a class Hourly that adds data members wage (which is the hourly wage) and hours (which is the number of hours worked that week). Write the appropriate member functions, including a printout() function prints out name, social security number, wage, hours and money earned that week (note: if hours > 40, the employee gets paid time and a half for the hours over 40).

Next, derive (from Employee) a class Salaried that adds a data member yearly salary. The printout() function prints out name, social security number, yearly salary, and salary for that week (which is yearly/52).