다음 문제에 대해 프로그램을 작성하고 해당 프로그램 스크립트(script) 파일을 제출한다. 프로그램은 주석(#) 으로 시작하고 첫째 주석으로 문제 번호, 본인의 학번과 이름을 반드시 표시한다. 스크립트 파일의 이름은 문제의 번호로 한다 (예 : 2번 문제의 스크립트파일 이름은 2.py). *수업에서 배우지 않은 함수는 사용하지 않는다. 단 현재까지 배운 자료형(int,float,complex,str,list,tuple,set,dictionary)의 모든 메소드함수를 사용할 수 있다*. 클래스룸에 제출 할 때는 연습문제의 모든 스크립트 파일을 ZIP으로 묶어 제출하며 ZIP 파일의 이름은 학번으로 설정한다. *지시사항대로 안 할 경우 감점이 될 수 있다.*
제출하는 스크립파일에 작성한 코드와 함께 예시로 나온 main()함수와 main()도 포함한다.

1. 일반적인 2차원 도형을 나타내는 Shape 클래스를 완성하고 이것을 상속받아서 Circle 클래스를 정의하여 다음 프로그램을 완성해보자. 출력은 오른쪽 밑의 박스와 같다. Shape 클래스는 color와 filled 속성을 가진다. Circle 클래스는 반지름을 나타내는 radius 변수를 추가로 정의한다.

```
class Shape:
    def __init__(self, color, filled):
        self.__color = color
        self.__filled = filled

    def __str__(self):
        return f'({self.__color},{self.__filled})'

class Circle(Shape):
```

```
def main() :
    a = Shape()
    b = Shape("red")
    print(a,b)
    c = Circle("blue",False,10)
    print(c)
    print(c.area())
main()
```

```
(yellow,True) (red,True)
(blue,False)(radius = 10)
314.0
```

2. 위의 Circle 클래스와 같이 Shape을 상속하여 Rectangle 클래스를 정의하는 프로그램을 하여라. Rectangle 클래스 width, height 변수를 추가로 정의한다. 메인함수와 출력은 다음과 같다.

```
def main() :
    c = Rectangle("blue",False,10,20)
    print(c)
    print(c.area())
main()
```

```
(blue,False)(10,20)
200
```

3. 다음과 같은 CPoint 클래스가 있다. CPoint에는 변수로 좌표를 나타내는 x 와 y 가 있다. 멤버함수로는 move(a,b)로 CPoint 객체의 x,y 좌표를 각각 a,b 만큼 이동한다. (move()의 return 값에 유의한다)
메인함수와 출력이 다음과 같이 되도록 CPoint 클래스를 프로그램 하라

```
def main() :
    p = CPoint()
    q = CPoint(3,4)
    print(p,q)
    print(q.move(-4,5))
    print(p.move(5,6).move(2,3))
main()
```

```
pos(0,0) pos(3,4)
pos(-1,9)
pos(7,9)
```

4. 클래스 Shape이 클래스 CPoint를 상속한다. class Shape(CPoint) (is-a 관계가 아니지만 상속해 본다). 메인함수와 출력이 다음과 같이 나오도록 Shape 클래스를 프로그램 하라. CPoint, Circle, Rectangle 클래스는 위 문제에서 프로그램 한 대로 그대로 사용한다.

```
def main() :
    a = Shape()
    b = Shape("red")
    c = Shape("black",False,1,2)
    print(a)
    print(b)
    print(c)
    a.move(2,3)
    print(a)
    print(b.move(4,5))
    d = Circle("blue",False,10).move(3,4)
    print(d)
    e = Rectangle("blue",False,10,20)
    print(e.move(7,8))
main()
```

```
pos(0,0)(yellow,True)
pos(0,0)(red,True)
pos(1,2)(black,False)
pos(2,3)(yellow,True)
pos(4,5)(red,True)
pos(3,4)(blue,False)(radius = 10)
pos(7,8)(blue,False)(10,20)
```

5. 클래스 Shape이 클래스 CPoint의 속성을 갖는다(has-a 관계). 클래스 Shape에 좌표 CPoint를 나타내는 cpoint 라는 변수를 추가한다(Shape 은 CPoint를 상속하지 않는다). 메인함수와 출력이 다음과 같이 나오도록 Shape 클래스를 프로그램 하라. CPoint, Circle, Rectangle 클래스는 위 문제에서 프로그램 한 대로 그대로 사용한다.

```
def main() :
    a = Shape()
    b = Shape("red")
    cpoint = CPoint(4,7)
    c = Shape("black",False,cpoint)
    print(a)
    print(b)
    print(c)
    a.move(2,3)
    print(a)
    print(b.move(4,5))
    d = Circle("blue",False,10).move(3,4)
    print(d)
    e = Rectangle("blue",False,10,20)
    print(e.move(7,8))
main()
```

```
pos(0,0)(yellow,True)
pos(0,0)(red,True)
pos(4,7)(black,False)
pos(2,3)(yellow,True)
pos(4,5)(red,True)
pos(3,4)(blue,False)(radius = 10)
pos(7,8)(blue,False)(10,20)
```

6. 다음과 같은 다중 상속에서 Child 클래스에서 Parent1, Parent2 클래스를 상속하고 있다. 상속의 순서가 Parent1, Parent2 인 경우 문제가 발생하지 않지만 보기에서 보여 주 듯이 Parent2, Parent1 으로 바뀌면 Child 클래스의 __init__() 함수의 super().__init__(a,b)에서 문제가 발생한다. Child 클래스의 상속 순서는 변경하지 말고 Parent1, Parent2 의 __init__() 함수를 수정하여 문제를 해결하여라. 수정 후 출력은 마지막 박스와 같이 된다

```
# Defining parent class 1
class Parent1 :
    def __init__(self,a,b) :
        super().__init__(b)
        self.p1 = a

    # Parent's show method
    def show(self):
        print("Inside Parent1",self.p1)
    def display(self) :
        print("Inside Parent1",self.p1)

 # Defining Parent class 2
class Parent2 :
    def __init__(self,b) :
        super().__init__()
        self.p2 = b
    # Parent's show method
    def display(self):
        print("Inside Parent2",self.p2)
```

```
# Defining child class
class Child(Parent2, Parent1):
    def __init__(self,a,b,c) :
        super().__init__(a,b)
        self.c = c

    # Child's show method
    def show(self):
        print("Inside Child",self.c)

# Driver's code
obj = Child(1,2,3)

obj.show()
obj.display()
```

```
Inside Child 3
Inside Parent2 2
```