한양대학교 ERICA 컴퓨터공학과 시험2 [총30점] 프로그래밍기초 2015년 2학기

반 : 목 / 금 **학번 :** **이름 :**

1. [3점] 컴퓨터과학이 여는 세계 / The Innovators

가. OX 문제 [다 맞아야 1점]	나.다음 중 기계학습과 관계 **없는** 하나는? [1점]
(1) 자연어로 표현할 수 있는 문제는 모두 컴퓨터로 풀 수 있다. (2) Python은 튜링완전(Turing-complete) 하다. (3) Python 프로그램은 모두 람다계산법(lambda calculus)으로 표현할 수 있다.	(1) 인덕 (induction) : 짐작해서 이끌기 (2) 앱덕 (abduction) : 원인 짐작하기 (3) 디덕 (deduction) : 반드시 이끌기

다. OX 문제 [다 맞아야 1점]

(1) 프로그램을 메모리에 저장한 뒤 실행하는 아키텍처를 von Neumann 아키텍처라고 하며, 이를 실현하여 최초로 판매용으로 제작한 컴퓨터는 UNIVAC 이다.

(2) 객체지향프로그램의 개념을 처음 도입한 언어는 Smalltalk 으로 Xerox PARC 연구소에서 Alan Kay가 중심이 되어 설계하고 구현하였다.

(3) GUI 개념은 Xerox PARC에서 처음 발생했지만, 실제로 상용 컴퓨터에 도입한 사례는 Apple 사의 Macintosh가 최초이다.

2. [3점] 중첩된 정수리스트를 받아서 안에 들어 있는 정수의 개수를 세어서 내주는 `countnumber` 함수를 작성하시오. 중첩된 정수리스트란 리스트의 원소가 정수 또는 정수리스트가 될 수 있는 리스트를 말한다. 즉, 아래 표의 왼쪽 열은 중첩된 정수리스트의 사례이고, 오른쪽 열은 각각에 `countnumber` 함수를 적용한 결과이다. 함수 `countnumber`를 작성하시오.

xs	countnumber(xs)
`[1,2,3]`	3
`[1,[],3]`	2
`[1,[1,2,[3,4]]]`	5
`[[[[[[[[1,2,3]]]]]]]]`	3
`[]`	0
`[[[[3]],[4]],5,6,[7]]`	5

`[1,`
` [2,2],`
` [[3],[4,4]],`
` [[[5,5,5,5]]],`
` 6,`
` [7,[[8],[[9]]]]]`
	14

귀띔: 원소가 리스트인지 아닌지 확인하고 싶으면 라이브러리 함수 `isinstance`를 다음과 같이 호출하면 알 수 있다.

```
isinstance(xs,list)
```

`xs`가 리스트이면 `True`를 내주고, 그렇지 않으면 `False`를 내준다.

정방행렬(square matrix)은 행과 열의 개수가 같은 다음과 같은 행렬을 말한다.

n_{00}	n_{01}	n_{02}	n_{03}
n_{10}	n_{11}	n_{12}	n_{13}
n_{20}	n_{21}	n_{22}	n_{23}
n_{30}	n_{31}	n_{32}	n_{33}

이 정방행렬은 리스트의 리스트로 다음과 같이 표현할 수 있다.

$[[n_{00}, n_{01}, n_{02}, n_{03}], [n_{10}, n_{11}, n_{12}, n_{13}], [n_{20}, n_{21}, n_{22}, n_{23}], [n_{30}, n_{31}, n_{32}, n_{33}]]$

3. [3점] 행렬(리스트의 리스트)을 인수로 받아서 정방행렬인지 확인하는 함수 `issquare`를 작성하시오. 즉, 정방행렬이면 `True`를 리턴하고, 그렇지 않으면 `False`를 리턴하면 된다. 예를 들어, 다음은 정방행렬이고,

```
[]
[[1]]
[[1,1],[1,1]]
[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]
```

다음은 정방행렬이 아니다.

```
[[]]
[[1,1],[1]]
```

4. [3점] 행렬의 행과 열을 바꾸는 걸 전치(transpose)라고 한다. 위의 정방행렬을 전치하면 다음과 같이 된다.

n_{00}	n_{10}	n_{20}	n_{30}
n_{01}	n_{11}	n_{21}	n_{31}
n_{02}	n_{12}	n_{22}	n_{32}
n_{03}	n_{13}	n_{23}	n_{33}

정방행렬을 받아서 이를 전치하여 내주는 함수 `transpose`를 작성하시오. (정방행렬에 대해서만 제대로 작동하면 된다. 즉, 정방행렬이 아닌 경우 오류가 나거나 이상한 결과가 나와도 괜찮다.)

5. [3점] 모든 행번호 i와 열번호 j에 대해서, $n_{ij} == -n_{ji}$ 를 만족하는 행렬을 반대칭행렬(antisymmetric matrix) 이라고 한다. 임의의 크기의 정방행렬을 인수로 받아서 반대칭행렬인지 아닌지 True 또는 False로 답해주는 함수 antisymmetric를 작성하시오. (정방행렬에 대해서만 제대로 작동하면 된다. 즉, 정방행렬이 아닌 경우 오류가 나거나 이상한 결과가 나와도 괜찮다.)

6. [3점] 정수 리스트를 받아서 정수 원소의 가능한 모든 순열(permutation)의 리스트를 만드는 함수 perm를 작성하시오. 실행사례는 다음 표와 같아야 한다.

xs	perm(xs)
[]	[[]]
[1]	[[1]]
[1,2]	[[1,2],[2,1]]
[1,2,3]	[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
[1,2,3,4]	[[1,2,3,4],[1,2,4,3],[1,3,2,4],[1,3,4,2],[1,4,2,3],[1,4,3,2], [2,1,3,4],[2,1,4,3],[2,3,1,4],[2,3,4,1],[2,4,1,3],[2,4,3,1], [3,1,2,4],[3,1,4,2],[3,2,1,4],[3,2,4,1],[3,4,1,2],[3,4,2,1], [4,1,2,3],[4,1,3,2],[4,2,1,3],[4,2,3,1],[4,3,1,2],[4,3,2,1]]

7. [3점] 숫자문자('0'-'9')로만 구성된 문자열에서 숫자의 빈도수를 세서 숫자와 빈도수 튜플의 리스트를 만들고, 빈도수가 높은 순으로 정렬하여 내주는 함수 digit_freq 를 만드시오.

xs	digit_freq(xs)
""	[('0', 0), ('1', 0), ('2', 0), ('3', 0), ('4', 0), ('5', 0), ('6', 0), ('7', 0), ('8', 0), ('9', 0)]
"0987654321"	[('0', 1), ('1', 1), ('2', 1), ('3', 1), ('4', 1), ('5', 1), ('6', 1), ('7', 1), ('8', 1), ('9', 1)]
"3077437827467203 4827764362738473"	[('7', 9), ('3', 6), ('4', 5), ('2', 4), ('6', 3), ('8', 3), ('0', 2), ('1', 0), ('5', 0), ('9', 0)]

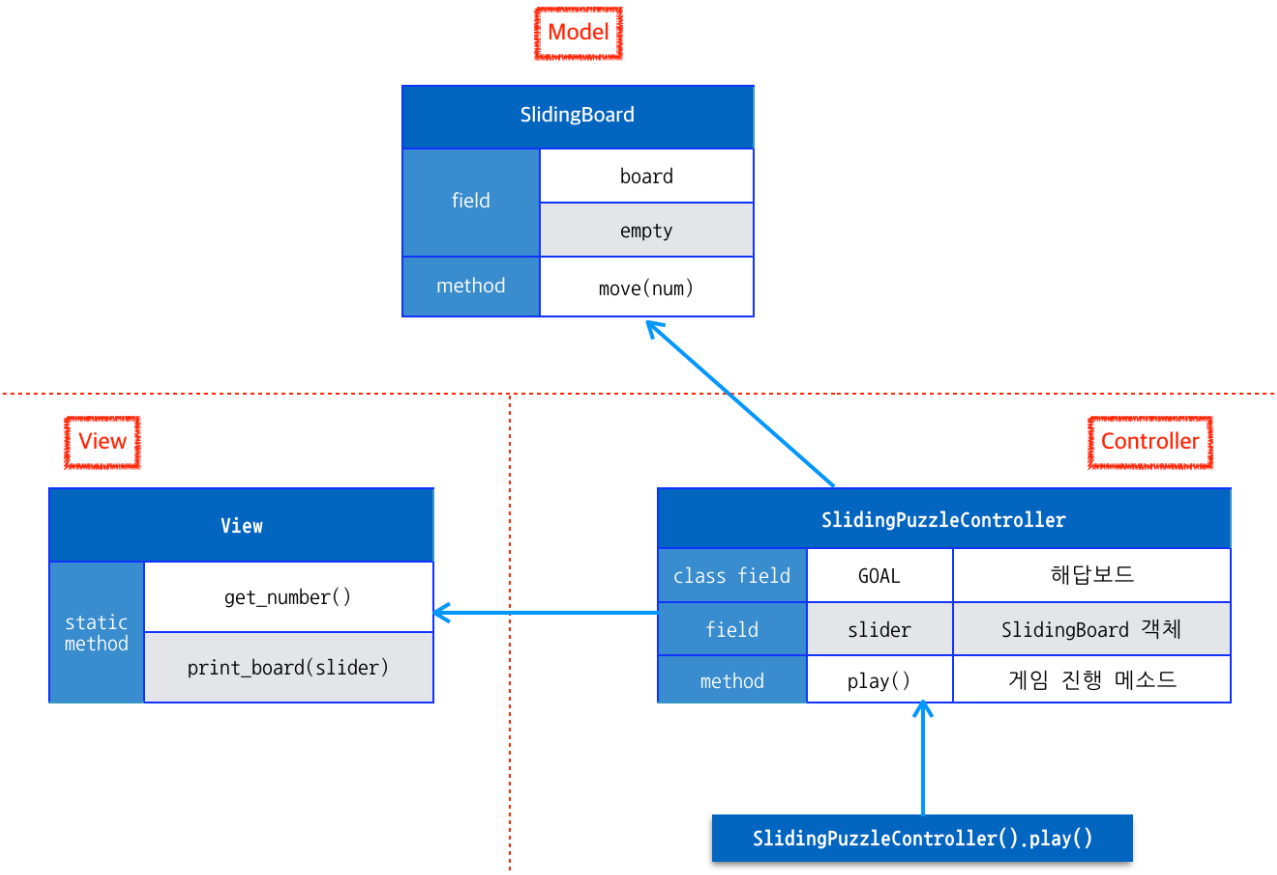
귀띔: 튜플의 리스트에서 튜플의 특정 원소를 키로 하여 역순으로 정렬하는 방법

```
>>> xs = [('4',24),('7',13),('6',17),('0',3)]
>>> xs.sort(key=lambda t: t[1], reverse=True)
>>> xs
[('4', 24), ('6', 17), ('7', 13), ('0', 3)]
```

8. [3점] 실습시간에 다음 표의 사양에 맞게 SlideBoard 클래스를 하나 만들어 슬라이드 퍼즐을 구현해보았다.

SlideBoard		
속성 (필드)	board	4 x 4 크기의 보드로 중첩리스트로 표현한다. 각 원소는 1~15까지의 수로 구성되며, 빈칸은 0으로 표시한다.
	empty	board에서 빈칸의 좌표로 i행, j열 위치는 (i,j)로 표현한다.
기능 (메소드)	move(num)	1~15 중 하나를 인수로 받아서 그 수가 빈칸에 인접해 있으면 그 수를 빈칸으로 옮긴다.
	print_board()	board를 다음과 같은 형식으로 실행창에 출력한다. S | 1 2 3 4 - + ---------- 1 | 13 8 1 2 | 10 14 9 7 3 | 6 3 5 4 4 | 2 12 11 15

잘 작동하는 코드는 나누어 준 "slidingpuzzle.py" 에 있다. 그런데 이 설계는 MVC 구조를 완전히 지키지 않았다. 다음의 MVC 설계 구조에 맞추어 이 프로그램을 수정하여 재구현 하시오. 수정한 클래스를 모두 하나의 파일에 넣어서 제출한다.



9. [3점] 학기 마지막 실습으로 공튀기기 애니메이션을 구현해보았다. MVC 구조에 기반하여 작성하였고 잘 작동하는 코드는 나누어 준 "bouncingball1.py"에 있다. 이 공튀기기 애니메이션에 파란색 공을 하나 추가하여 공 두개가 돌아다니도록 프로그램을 수정하시오. 빨간색 공의 초기 방향과 속도는 (2, 5)로 똑 같이 하고, 파란색 공의 초기방향과 속도는 (-5, -2)로 지정한다.

10. [3점] 9번 문제에서 두 공은 서로 만나더라도 그대로 진행한다. 두 공이 만나면, 서로의 진행 방향을 맞교환하여 진행하도록 코드를 수정하시오. 여기서 만난다는 의미는 공이 겹치기 시작하는 시점을 의미한다. (공의 만나는 지점을 치밀하게 계산하지 않았다고 틀리게 하지는 않는다.) 수정한 코드는 "bouncingball2.py" 에 저장하여 제출하시오.