1시간 40분 동안 진행하는 코딩 시험입니다. 주어진 뼈대 코드를 내려받아 코딩을 완성한 다음, 제출 마감 시간 안에 파일을 업로드 하세요**. (5문제, 3쪽)**

시험 보는 동안 타인과 소통 시도는 부정행위로 간주합니다. 컴퓨터에 소통을 위한 창을 켜는 것 자체도 부정행위로 간주하고 금합니다. 부정행위로 적발되는 경우 F 처리하고, 대학 본부에 보고합니다.

문제 1. [14점] 전자개표기 (7점/문제)

익명으로 인터넷 찬반투표를 실시하여 투표함에 찬성은 "O", 반대는 "X"로 표시하여 리스트로 다음과 같이 모은다고 하자.

```
box0 = ['X','O','O','X','O','O','O','O','O','O',
        'O','O','X','O','X','X','O','X','X','X',
        'X','X','X','O','O','X','O','X']
box1 = ['X','O','O','X','O','O','X','O','O','X',
        'O','O','X','O','X','X','O','X','X','X',
        'X','X','X','O','O','X','O','X']
```

개표 결과를 실행창에 보여주는 함수는 다음과 같이 작성하였다.

```
def show_ballot_box(box):
    counter = 0
    for ballot in box:
        print(ballot, end=' ')
        counter += 1
        if counter % 10 == 0:
            print()
    if counter % 10 != 0:
        print()
```

이 함수를 위의 투표함 box0와 box1을 인수로 각각 실행하여 실행창에 어떻게 나타나는지 확인하자.

이 투표함을 개표하는 함수는 다음과 같이 작성할 수 있다.

```
def ballot_sorter(box):
    yes = 0
    no = 0
    o_box = []
    x_box = []
    for ballot in box:
        if ballot == "O":
            yes += 1
            o_box.append(ballot)
        else: # ballot == "X"
            no += 1
            x_box.append(ballot)
    print("개표 결과 (정상)")
    print("찬성 =", yes)
    show_ballot_box(o_box)
    print("반대 =", no)
    show_ballot_box(x_box)
```

이 함수를 위 투표함을 인수로 하여 호출하면 다음과 같이 실행창에 개표 결과가 나타난다.

ballot_sorter(box0) 호출 결과

```
개표 결과 (정상)
찬성 = 15
O O O O O O O O O O
O O O O O
반대 = 13
X X X X X X X X X X
X X X
```

ballot_sorter(box1) 호출 결과

```
개표 결과 (정상)
찬성 = 13
O O O O O O O O O O
O O O
반대 = 15
X X X X X X X X X X
X X X X X
```

인터넷 익명 투표 소프트웨어 시스템이 개표 프로그램의 단순한 논리 변경 해킹으로 인한 오작동에 얼마나 취약한지 확인시켜주기 위한 목적으로, 개표 결과를 항상 가결되도록 조작하는 개표 함수를 두 가지 만들어보자. 찬성과 반대가 동률인 경우 부결된 것으로 한다.

(1) 반대표의 기표수가 반(1/2)에 도달하기 직전까지는 제대로 분류하다가, 그 이후에는 반대표를 모두 찬성표로 분류하는 함수 ballot_sorter_rig1을 위의 ballot_sorter의 일부를 고쳐서 작성하자.

```
def ballot_sorter_rig1(box):
    pass
```

실행 사례

```
>>> ballot_sorter_rig1(box0)
개표 결과 (조작1)
찬성 = 15
O O O O O O O O O O
O O O O O
반대 = 13
X X X X X X X X X X
X X X

>>> ballot_sorter_rig1(box1)
개표 결과 (조작1)
찬성 = 15
O O O O O O O O O O
O O O O O
반대 = 13
X X X X X X X X X X
X X X
```

(2) 반대표가 n장 나올 때마다, 반대표 1장을 찬성표로 분류하는 함수 ballot_sorter_rig2를 위의 ballot_sorter의 일부를 고쳐서 작성하자. 여기서 n은 추가 인수로 제공한다. 이 경우 n 값을 너무 크게 잡으면 조작에 실패할 수도 있다.

```
def ballot_sorter_rig2(box, n):
    pass
```

실행 사례

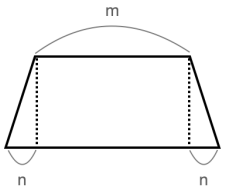
```
>>> ballot_sorter_rig2(box0,10)
개표 결과 (조작2)
찬성 = 16
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
반대 = 12
X X X X X X X X X X
X X

>>> ballot_sorter_rig2(box1,10)
개표 결과 (조작2)
찬성 = 14
0 0 0 0 0 0 0 0 0 0
0 0 0 0
반대 = 14
X X X X X X X X X X
X X X X

>>> ballot_sorter_rig2(box1,5)
개표 결과 (조작2)
찬성 = 16
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
반대 = 12
X X X X X X X X X X
X X
```

문제 2. [7점] ASCII 아트

자연수 m, n을 인수로 받아서 실행창에 등변사다리꼴을 아래의 실행 사례와 같이 프린트하는 함수 iso_trapezoid를 작성하시오. 자연수 m은 윗변의 길이이고, n은 아래 그림과 같이 아랫변의 증가폭을 나타낸다.



실행 사례

```
>>> iso_trapezoid(3,5)
     000
    00000
   0000000
  000000000
 00000000000
0000000000000

>>> iso_trapezoid(4,2)
  0000
 000000
00000000

>>> iso_trapezoid(8,5)
     00000000
    0000000000
   000000000000
  00000000000000
 0000000000000000
000000000000000000

>>> iso_trapezoid(9,3)
   000000000
  00000000000
 0000000000000
000000000000000
```

문제 3. [14점] 스스로 오목 (7점/문제)

오목은 바둑판에 두 사람이 번갈아 돌을 놓아 가로나 세로, 대각선으로 다섯 개 연속된 돌을 놓으면 이기는 놀이이다. 미니 바둑판(9 x 9)에 프로그램이 스스로 오목을 무작위로 두는 프로그램을 만들어보자. 프로그램에서 흰돌은 "O"로 검은돌은 "X"로 표현하고, 빈 자리는 빈문자열 " "로 표현한다. 바둑판의 상태는 실행창에 다음과 같이 2차원으로 보여 준다.

```
O   X O     X O O
X       O   X   X
X X   X X   X   X
X   O O   X O X O
O O O X   X O X O
X O O X X X   X
O X X X O O O O O
  O X O X   O O X
  O X O         O
```

흰돌이 오목이 되어서 막 오목놀이가 끝난 상태이다. 이 바둑판은 프로그램에서 다음과 같이 2차원 리스트로 표현한다.

```
b0 = [['O',' ','X','O',' ',' ','X','O','O'],
      ['X',' ',' ',' ','O',' ','X',' ','X'],
      ['X','X',' ','X','X',' ','X',' ','X'],
      ['X',' ','O','O',' ','X','O','X','O'],
      ['O','O','O','X',' ','X','O','X','O'],
      ['X','O','O','X','X','X',' ','X',' '],
      ['O','X','X','X','O','O','O','O','O'],
      [' ','O','X','O','X',' ','O','O','X'],
      [' ','O','X','O',' ',' ',' ',' ','O']]
```

다음 함수를 show_go_board(b0)과 같이 호출하면 위와 같은 모양으로 실행창에 보여줄 수 있다.

```
def show_go_board(b):
    size = len(b)
    for i in range(size):
        for j in range(size):
            print(b[i][j],end=" ")
        print()
```

(1) 바둑판 리스트를 인수로 받아서, 오목이 되어 승부가 결정되었는지 확인하는 함수 check_omog을 주어진 뼈대코드를 채워서 작성하자. 이 함수는 오목이면 True, 그렇지 않으면 False를 리턴한다. 테스트 데이터로 오목이 되는 바둑판 4개, 오목이 되지 않는 바둑판 4개를 뼈대 코드에 제공한다.

(2) check_omog 함수가 완성이 되면, 승패가 결정될 때까지 프로그램이 스스로 오목놀이를 하는 함수 play_omog을 작성하자. 검은돌을 항상 먼저 두도록 하고, 둘 곳은 무작위로 선택하며, 오목이 되어 놀이가 끝나면 누가 몇 수 만에 이겼는지 다음과 같이 바둑판과 함께 보여주어야 한다.

```
O O X
X X X     X
    X   O X   O
    O X   O O O
      X X       O
X     X   X   O
X   O   X     O O
O     X       O O
    X X   O X O
X wins in 37 tries
```

빈 자리를 다 채웠음에도 불구하고 오목이 되지 않으면 바둑판을 보여주면서 무승부임을 다음과 같이 알려주어야 한다.

```
O X X O O O X O O
O X O X O O X X X
X X O X O X X X O
X O O X X O X X X
X O X O X O O O O
X X O X X X O X O
O X X O O O O X O
X O O O X O O X O
X O O O X O X X X
No winners!
```

빈 바둑판을 만드는 다음 함수는 뼈대 코드에서 제공되며, 편리한 대로 써도 좋다.

```
def initialize_board():
    board = [[] for _ in range(9)]
    for i in range(9):
        for _ in range(9):
            board[i].append(" ")
    return board
```