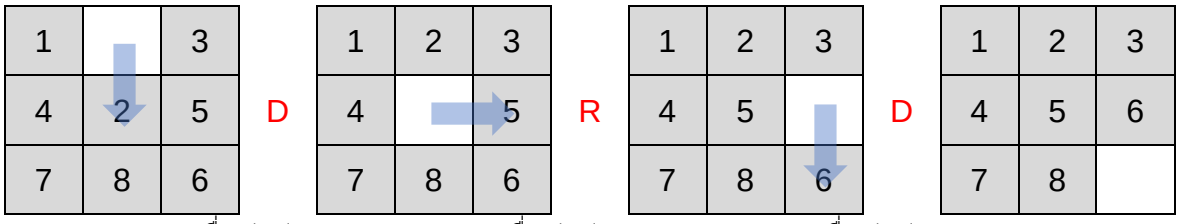
Assignment 6: 8-Puzzle

คงเคยเล่นเกม 15 puzzle กันมาก่อน เป็นแผ่นพลาสติกรูปสี่เหลี่ยมจัตุรัส ภายในประกอบด้วยแผ่นพลาสติกย่อยเล็ก ๆ (สี่เหลี่ยมจัตุรัสเหมือนกัน) จำนวน 15 แผ่น (แต่ละแผ่นมีตัวเลขกำกับตั้งแต่ 1 ถึง 15) วางเรียงกันเป็นตาราง 4×4 โดยมีช่องว่างหนึ่งช่องอยู่ภายใน สามารถเลื่อนแผ่นต่าง ๆ ไปมาได้ในแนวนอนแนวตั้งเมื่ออยู่ติดกับช่องว่าง เช่น ในรูปทางขวานี้ สามารถเลื่อนหมายเลข 8 ลง หรือเลื่อน 1 ขึ้น หรือเลื่อน 5 ซ้าย วัตถุประสงค์ของการเลื่อนคือ เลื่อนให้หมายเลขในกระดานเรียงลำดับ 1,2,3,... จากซ้ายไปขวาทีละแถวทีละแถว และให้ช่องมุมล่างขวาเป็นช่องว่าง (https://en.wikipedia.org/wiki/15_puzzle)

แต่ในโจทย์ข้อนี้ แทนที่จะบอกว่าเลื่อนหมายเลข จะขอบอกเป็นเลื่อนช่องว่างแทน ดังนั้นในรูปบนนี้ สามารถเลื่อนช่องว่าง ขึ้น ลง และขวาได้ (เพื่อความกระชับ ขอใช้ตัวอักษร U, D, L และ R แทนการเลื่อนช่องว่างขึ้น ลง ซ้าย และขวาตามลำดับ)

โจทย์ข้อนี้เกี่ยวกับโปรแกรมหาคำตอบของการเลื่อนช่องว่างจนได้กระดานสุดท้ายที่ต้องการ แต่จะไม่ทำกับกระดาน 4x4 ขอทำแค่ 3x3 (เรียกว่า 8 Puzzle) แต่ละกระดานแทนด้วยลิสต์ที่เก็บหมายเลข 0 ถึง 8 ลำดับหมายเลขในลิสต์เรียงตามหมายเลขในกระดานจากซ้ายไปขวาทีละแถว โดยที่ 0 แทนช่องว่าง ดูตัวอย่างข้างล่างนี้



งานของคุณ

(ดูโปรแกรมในหน้าถัดไปประกอบ) จงเติมชุดคำสั่งในบริเวณสีเขียว ตามข้อกำหนดดังนี้

- มีตัวแปร `node` (มีมาให้แล้ว ใช้ได้เลย) เป็นลิสต์ เก็บข้อมูล 10 ตัว
  - เก้าตัวแรกคือ ลำดับของหมายเลข 0 ถึง 8 แทนหมายเลขในกระดาน
  - ตัวสุดท้ายเป็นสตริง เก็บลำดับการย้ายช่องว่างของอดีตที่ผ่านมา (ไม่ต้องสนใจว่า สตริงนี้เก็บอะไรอยู่)
- ภาระที่ต้องทำคือ ให้สร้างลิสต์ใหม่ชื่อ `successors` ภายในเก็บข้อมูลจำนวน 10M ตัว โดยที่
  - `M` คือจำนวนกระดานใหม่ที่ได้จากการเลื่อนช่องว่างของ `node` ในทิศทางต่าง ๆ ที่ทำได้, $2 \le M \le 4$
    เช่น ถ้าช่องว่างใน `node` อยู่ที่มุม ก็เลื่อนช่องว่างได้แค่สองทาง `M = 2`
  - เลข 10 (ของ 10M) คือข้อมูลสิบตัวที่มีความหมายเหมือนกับใน `node`
    - เก้าตัวแรกคือ ลำดับของหมายเลข 0 ถึง 8 แทนหมายเลขในกระดานใหม่ที่ได้จากการเลื่อนช่องว่างของ `node`
    - ตัวสุดท้ายเป็นสตริง มีค่าเท่ากับตัวสุดท้ายของ `node` ต่อด้วย ตัวอักษร `U`, `D`, `L` หรือ `R` ขึ้นกับว่าเป็นการเลื่อนช่องว่างของ `node` ขึ้น ลง ซ้าย หรือ ขวาตามลำดับ

ตัวอย่างเช่น (ดูรูปข้างล่างนี้) หาก node มีค่าเท่ากับ `[1,0,3,4,2,5,7,8,6,'LR']` (รูปซ้าย) ซึ่ง 0 อยู่แถวบนสุด จึงเลื่อนได้แค่ ลง, ซ้าย และ ขวา ได้ผลในแต่ละกรณีของการเลื่อนช่องว่างคือ รูปกลาง (3 แบบ) เมื่อรวมทั้ง 3 แบบ ได้ลิสต์ผลลัพธ์ทางขวา

successors = [

1,2,3,4,0,5,7,8,6,'LRD',

0,1,3,4,2,5,7,8,6,'LRL',

1,3,0,4,2,5,7,8,6,'LRR'

]

```
# Prog-06: 8-Puzzle
# Fill in your ID & Name
# ...
# Declare that you do this by yourself

def is_goal(node):
    return node[:-1] == \
           list(range(1,len(node)-1))+[0]

def insert_all(node, fringe):
    n = len(node)
    children = gen_successors(node)
    for i in range(0,len(children),n):
        fringe.append(children[i:i+n])

def bfs(board):
    start_node = board + ['']
    fringe = [start_node]
    while True:
        if len(fringe) == 0:
            break
        front = fringe[0]
        fringe = fringe[1:]
        if is_goal(front):
            return front[-1]
        insert_all(front,fringe)
    return ''

def print_successors(s):
    N = 1
    for e in s:
        if type(e) is str: break
        N += 1
    for i in range(0,len(s),N):
        print(s[i:i+N])
#----------------------------------------
```

download code นี้ได้

ไม่เปลี่ยนแปลงคำสั่งสีแดงเด็ดขาด

```
def gen_successors(node):
    successors = []
    return successors
#----------------------------------------
def print_moves(board, moves):
    # bonus function: optional
    return
#----------------------------------------
board = [4,1,3,2,5,6,7,8,0]  # change this list
s = gen_successors(board + ['UDR'])
print_successors(s)
moves = bfs(board)
print(moves)
print_moves(board, moves)
```

ตัวอย่าง

```
node = [1,0,3,4,2,5,7,8,6,'LR']
```

ก็ต้องสร้าง

```
successors = \
       [1,2,3,4,0,5,7,8,6,'LRD',
        0,1,3,4,2,5,7,8,6,'LRL',
        1,3,0,4,2,5,7,8,6,'LRR'
       ]
```

ตัวอย่าง

```
board = [1,0,3,4,2,5,7,8,6]
moves = 'DRD'
```

ก็ต้องแสดงผลลัพธ์ทางจอภาพดังนี้

```
 1  0  3
 4  2  5
 7  8  6
--------- D
 1  2  3
 4  0  5
 7  8  6
--------- R
 1  2  3
 4  5  0
 7  8  6
--------- D
 1  2  3
 4  5  6
 7  8  0
```

ตัวอย่าง

ค่าของลิสต์ board ตอนเริ่มโปรแกรม	output (ทางจอภาพ)

`[4, 1, 3, 2, 5, 6, 7, 8, 0]`	`[4, 1, 3, 2, 5, 6, 7, 0, 8, 'UDRL']`
	`[4, 1, 3, 2, 5, 0, 7, 8, 6, 'UDRU']`
	`LULURDDR`

`[1, 3, 6, 4, 0, 2, 7, 5, 8]`	`[1, 0, 6, 4, 3, 2, 7, 5, 8, 'UDRU']`
	`[1, 3, 6, 4, 5, 2, 7, 0, 8, 'UDRD']`
	`[1, 3, 6, 0, 4, 2, 7, 5, 8, 'UDRL']`
	`[1, 3, 6, 4, 2, 0, 7, 5, 8, 'UDRR']`
	`RULDDR`

`[2, 3, 6, 0, 1, 5, 4, 7, 8]`	`[0, 3, 6, 2, 1, 5, 4, 7, 8, 'UDRU']`
	`[2, 3, 6, 4, 1, 5, 0, 7, 8, 'UDRD']`
	`[2, 3, 6, 1, 0, 5, 4, 7, 8, 'UDRR']`
	`RRULLDDRR`

หมายเหตุ: ผลลัพธ์ที่ได้จากโปรแกรมที่ทำงานถูกต้องอาจมีลำดับที่ไม่เหมือนกับที่แสดงข้างบนนี้ก็ได้ เพราะผลลัพธ์ขึ้นกับลำดับของการสร้าง U, D, L, R ตอนสร้าง successors ในบริเวณสีเขียว

โบนัส

- ชุดคำสั่งที่เขียนในบริเวณสีเขียว สามารถรับกระดานขนาด 3x3 และอื่น ๆ ได้ ทั้ง 2x2, 4x4, 5x5, ..., NxN (N เป็นจำนวนเต็มมากกว่าเท่ากับ 2)
- เขียนชุดคำสั่งในบริเวณสีเหลือง (เพื่อแสดงลำดับการเปลี่ยนแปลงกระดานจะเริ่มต้นจนได้คำตอบ)
  - มีตัวแปรสองตัวข้างล่างนี้ให้แล้ว
    - `board` เป็นลิสต์ขนาด N^2 ช่อง ภายในเก็บหมายเลข 0 ถึง (N^2 – 1) ในกระดานขนาด NxN
    - `moves` เป็นสตริง ที่ภายในมีตัวอักษร U, D, L หรือ R เท่านั้น (มีได้หลาย ๆ ตัว หรือไม่มีสักตัวก็ได้)
  - สิ่งที่ต้องทำคือ แสดงรายละเอียดข้อมูลในกระดาน เริ่มจาก `board` ที่ได้รับ และก็แสดงกระดานใหม่ต่าง ๆ ที่ได้จากการเลื่อนช่องว่างตามทิศทางของตัวอักษรใน `moves`
  - เช่น `board` มีค่า `[1,0,3,4,2,5,7,8,6]` และ `moves = 'DRD'` จะได้ผลลัพธ์ที่แสดงทางจอภาพเป็นดังแสดงข้างล่างนี้

```
 1  0  3
 4  2  5
 7  8  6
--------- D
 1  2  3
 4  0  5
 7  8  6
--------- R
 1  2  3
 4  5  0
 7  8  6
--------- D
 1  2  3
 4  5  6
 7  8  0
```

หมายเหตุ

โปรแกรมของงานนี้ใช้วิธีที่เรียกว่า Breadth-first search (https://en.wikipedia.org/wiki/Breadth-first_search) แบบที่ไม่ได้เติมลูกเล่นอะไรเลยในการหาคำตอบ จึงใช้ได้เฉพาะกับกระดานง่าย ๆ ของ 8-puzzle เท่านั้น (ถ้านำไปใช้กับกระดานยาก เช่นกรณีข้างล่างนี้ หรือกระดานที่ใหญ่ขึ้น เช่น 15-puzzle มักจะหาคำตอบไม่สำเร็จ)