Assignment #5 64015030

ชินพัฒน์ ลิ้มประธาน

1. เขียนโปรแกรมเพื่อแสดงปิรามิดบนจอภาพ โดยแสดงเป็นรูปสี่เหลี่ยม ตัวเลขเริ่มจาก 0 และเพิ่มค่าตามลำดับจำนวนขั้น จาก 0 ถึง N (รับค่า N จากคีย์บอร์ด) ค่าของ N อยู่ระหว่าง 0-9 ดังตัวอย่าง

```
                                        0000000
                            00000       0111110
            000             01110       0122210
0           010             01210       0123210
            000             01110       0122210
                            00000       0111110
                                        0000000

n=0         n=1             n=2         n=3
```

```python
while True:
    n = int(input("Enter n : (0-9): "))
    if n in range(0, 10):
        break
    else:
        print("Please enter number between 0 and 9! :")

for i in range(0,n+1):
    for j in range(0, n+1):
        if i ≤ j:
            print(i, end='')
        else:
            print(j, end='')
    for k in range(n-1, -1, -1):
        if i ≤ k:
            print(i, end="")
        else:
            print(k, end="")
    print("")
for i in range(n-1, -1, -1):
    for j in range(0, n+1):
        if i ≥ j:
            print(j, end="")
        else:
            print(i, end="")
    for k in range(n-1, -1, -1):
        if i ≥ k:
            print(k, end="")
        else:
            print(i, end="")
    print("")
```

```
Enter n : (0-9): 0
0
```

```
Enter n : (0-9): 1
000
010
000
```

```
Enter n : (0-9): 2
00000
01110
01210
01110
00000
```

```
Enter n : (0-9): 4
000000000
011111110
012222210
012333210
012343210
012333210
012222210
011111110
000000000
```

```
Enter n : (0-9): 3
0000000
0111110
0122210
0123210
0122210
0111110
0000000
```

2. เขียนโปรแกรมเพื่อแสดงปิรามิดบนจอภาพ โดยแสดงเป็นรูปสี่เหลี่ยม ตัวเลขเริ่มจาก 0 และเพิ่มค่าตามลำดับจำนวนขั้น จาก 0 ถึง N (รับค่า N จากคีย์บอร์ด)ค่าของ N อยู่ระหว่าง 0-9 ดังตัวอย่าง

```
                                                0
                                       0       010
                          0           010     01210
        0                010          01210  0123210
                          0           010     01210
                                       0       010
                                                0

        n=0              n=1          n=2     n=3
```

```python
while True:
    n = int(input("Enter n : (0-9): "))
    if n in range(0, 10):
        break
    else:
        print("Please enter number between 0 and 9! :")

for i in range(0, n+1):
    for s in range(0, n-i):
        print(" ", end="")
    for j in range(0, i+1):
        print(j, end="")
    for k in range(i-1, -1, -1):
        print(k, end="")
    print("")
for i in range(0, n):
    for s in range(0, i+1):
        print(" ", end="")
    for j in range(0, n-i):
        print(j, end="")
    for k in range(n-i-2, -1, -1):
        print(k, end="")
    print("")
```

```
Enter n : (0-9): 0
0
```

```
Enter n : (0-9): 3
   0
  010
 01210
0123210
 01210
  010
   0
```

```
Enter n : (0-9): 2
  0
 010
01210
 010
  0
```

3. กล่องขนาด 6 x 6 ช่อง แต่ละช่องสามารถเก็บข้อมูลได้ 1 ตัวเท่านั้น หลักการใส่ข้อมูลคือ จะต้องใส่ข้อมูลจากด้านบนเท่านั้น ซึ่งข้อมูลจะหล่นลงไปที่ชั้นใดนั้นขึ้นกับมีข้อมูลเดิมอยู่ในช่องนั้นหรือไม่ ถ้าไม่มีข้อมูลอื่นอยู่เลยก็จะหล่นไปชั้นที่ 6 (ชั้นสุดท้าย) แต่ถ้ามีข้อมูลอื่นอยู่จะหล่นอยู่ชั้นถัดมา ตัวอย่างในรูป เช่นในช่องที่ 3 หากต้องการใส่ x ลงไปจะไปอยู่ที่พิกัด [4,3] (ชั้นที่ 4 ของช่อง 3) ให้นักศึกษาเขียนโปรแกรมเล่นเกม โดยผลัดกันใส่ข้อมูลใส่ลงในกล่องดังกล่าว ถ้ามีข้อมูลติดกัน 3 ตัวไม่ว่าแนวตั้ง แนวนอน หรือแนวเฉียงก่อนจะเป็นฝ่ายชนะ(คล้ายกับเกม o x) โดยนักศึกษาจะรับ input เป็นช่องที่ต้องการใส่ข้อมูล ส่วนเครื่องจะทำการ random ช่องที่จะใส่ 1 ใน 6 ช่อง การดำเนินการเล่นจะสิ้นสุดเมื่อมีฝ่ายใด ฝ่ายหนึ่งชนะ (ให้แสดงข้อมูลในกล่องทุกครั้งก่อนผู้เล่นจะใส่ข้อมูล และรายงานด้วยว่าใครเป็นผู้ชนะ) (comp : w, user : x)

1						
2						
3						
4						
5			w			
6			w	x	w	x
	1	2	3	4	5	6

ตัวอย่างการแสดงข้อมูลในกล่องที่หน้าจอ

```
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
Enter slot (1-6) : 2
```

```
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|x|w|o|o|o|
Enter slot (1-6) : 2
```

```
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|o|o|o|o|o|
o|x|w|o|o|o|
o|x|w|o|o|o|
```

```python
import random

BOX_SIZE = 6
TOP_ROW = 0
EMPTY_SLOTH = "_"
USER_COIN = "X"
COMP_COIN = "W"
boxs = []

def create_box():
    for _ in range(BOX_SIZE):
        boxs.append([EMPTY_SLOTH, EMPTY_SLOTH, EMPTY_SLOTH, EMPTY_SLOTH, EMPTY_SLOTH, EMPTY_SLOTH])

def draw_box():
    print(f"   {'1':^5} {'2':^5} {'3':^5} {'4':^5} {'5':^5} {'6':^5}")
    for row in range(BOX_SIZE):
        print(f"{row+1} |  ", end="")
        for column in range(len(boxs[row])):
            print(f"{boxs[row][column]}  |  ", end="")
        print("") # for empty spaces
    print("="*39)

def insert_coin(select_column, coin):
    is_slot_empty = True
    current_row = BOX_SIZE - 1
    while is_slot_empty:
        if current_row < TOP_ROW:
            break
        for column in range(BOX_SIZE):
            if (column == select_column) and boxs[current_row][column] == EMPTY_SLOTH:
                boxs[current_row][column] = coin
                is_slot_empty = False
                break
        current_row += -1
    if not is_slot_empty:
        return True
    else:
        return False
```

```python
###! search
def horizontal_search(coin):
    row = 0
    found = False
    while not found:
        if row > 5:
            break
        for column in range(len(boxs[row])-2):
            if boxs[row][column] is boxs[row][column+1] is boxs[row][column+2] == coin:
                print(f"{coin} was found on row {row+1} col {column+1}")
                found = True
                break
        row += 1
    return found

def vertical_search(coin):
    row = 0
    found = False
    while not found:
        if row > 3:
            break
        for column in range(0, len(boxs[row])):
            if boxs[row][column] is boxs[row+1][column] is boxs[row+2][column] == coin:
                print(f"{coin} was found on row {row+1} col {column+1}")
                found = True
                break
        row += 1
    return found
```

```python
def cross_search_right(coin):
    row = 0
    found = False
    while not found:
        if row > 3:
            break
        for column in range(len(boxs[row])-2):
            if boxs[row][column] is boxs[row+1][column+1] is boxs[row+2][column+2] == coin:
                print(f"{coin} was found on row {row+1} col {column+1}")
                found = True
                break
        row += 1
    return found

def cross_search_left(coin):
    row = 0
    found = False
    while not found:
        if row > 3:
            break
        for column in range(2, len(boxs[row])):
            if boxs[row][column] is boxs[row+1][column-1] is boxs[row+2][column-2] == coin:
                print(f"{coin} was found on row {row+1} col {column+1}")
                found = True
                break
        row += 1
    return found
###!
```

```python
def is_user_win():
    if horizontal_search(USER_COIN) or vertical_search(USER_COIN) or cross_search_right(USER_COIN) or cross_search_left(USER_COIN):
        return True
    else:
        return False

def is_com_win():
    if horizontal_search(COMP_COIN) or vertical_search(COMP_COIN) or cross_search_right(COMP_COIN) or cross_search_left(COMP_COIN):
        return True
    else:
        return False
```

```python
#! MAIN
create_box()
is_game_finished = False

while not is_game_finished:
    draw_box()
    correct_input = False
    while not correct_input:
        user_select = input(f"select column to drop (1-{BOX_SIZE}): ")
        try:
            if int(user_select) in range(BOX_SIZE+1) and insert_coin(int(user_select)-1, USER_COIN):
                break
            else:
                print(f"please choose column between 1-{BOX_SIZE}: ")
        except ValueError:
            print("please enter only number! ")
    print("\n")
    bot_select = random.randint(0,5)
    insert_coin(bot_select, COMP_COIN)

    if is_user_win():
        draw_box()
        print("You win !")
        is_game_finished = True
    elif is_com_win():
        draw_box()
        print("COMP WIN YOU LOSE !")
        is_game_finished = True

    if is_game_finished:
        play_again = input("Play Again? (Y/N) : ")
    else:
        play_again = 'n'

    if play_again.lower() == "y":
        is_game_finished = False
        boxs.clear()
        create_box()
        print("\n")
```

```
X was found on row 4 col 4
     1     2     3     4     5     6
1 |  _  |  _  |  _  |  _  |  _  |  _  |
2 |  _  |  _  |  _  |  _  |  _  |  _  |
3 |  _  |  _  |  W  |  _  |  _  |  _  |
4 |  _  |  _  |  W  |  X  |  _  |  _  |
5 |  _  |  W  |  X  |  X  |  _  |  _  |
6 |  X  |  X  |  W  |  W  |  _  |  _  |
=========================================
You win !
Play Again? (Y/N) : n
```

```
W was found on row 4 col 6
     1     2     3     4     5     6
1 |  _  |  _  |  _  |  _  |  _  |  _  |
2 |  W  |  _  |  _  |  _  |  _  |  _  |
3 |  X  |  _  |  X  |  _  |  _  |  X  |
4 |  X  |  _  |  W  |  _  |  _  |  W  |
5 |  W  |  _  |  X  |  _  |  W  |  X  |
6 |  X  |  _  |  W  |  W  |  X  |  W  |
=========================================
COMP WIN YOU LOSE !
Play Again? (Y/N) : n
```

4. ให้เขียนโปรแกรมค้นหาว่ามีข้อความ KMITL (เรียงติดกัน) บนตารางที่กำหนดให้กี่คำ พร้อมแสดงตำแหน่งของทุกตัวอักษรที่ประกอบกันเป็นข้อความ KMITL ของทุกคำ

นักศึกษาสามารถกำหนดค่าเริ่มต้นของตารางได้ดังนี้

```
List ['*****',
      '*****',
      '*****',
      '*****',
      '*****',
     ]
```

ตัวอย่าง

เมื่อกำหนดค่า Table เป็นดังนี้จะได้ผลลัพธ์คือ

*	*	*	*	*
*	M	M	*	*
*	K	I	K	*
*	I	T	*	*
*	*	L	*	*

K 3 2 M 2 2 I 3 3 T 4 3 L 5 3

K 3 2 M 2 3 I 3 3 T 4 3 L 5 3

K 3 4 M 2 3 I 3 3 T 4 3 L 5 3

KMITL Count = 3

```
string_container = []

#*                    ROW   0    1    2    3    4    COLUMN
string_container.append(['*', '*', '*', '*', '*']) #* 0
string_container.append(['*', 'M', 'M', '*', '*']) #* 1
string_container.append(['*', 'K', 'I', 'K', '*']) #* 2
string_container.append(['*', 'I', 'T', '*', '*']) #* 3
string_container.append(['*', '*', 'L', '*', '*']) #* 4

def draw_table():
    print(f"ROW/COL  {'1':^4} {'2':^4} {'3':^4} {'4':^4} {'5':^4}")
    for row in range(len(string_container)):
        print(f" {row+1:^5}{'':<2}{string_container[row]}")
    print("\n")

def search_around(current_row, current_column, char_finding):
    return_list = []
    for row in range(-1, 2):
        for col in range(-1, 2):
            if (string_container[current_row+row][current_column+col] == char_finding):
                return_list.append(f"{current_row+row}{current_column+col}")
    return return_list

k_list = []
result_list = []
```

```python
for row in range(len(string_container)):
    for column in range(len(string_container[row])):
        if string_container[row][column] == "K":
            k_list.append(f"{row}{column}")

for k in k_list:
    current_row = int(k[0])
    current_column = int(k[1])
    m_list = search_around(current_row, current_column, "M")

    for m in m_list:
        current_row = int(m[0])
        current_column = int(m[1])
        i_list = search_around(current_row, current_column, "I")

        for i in i_list:
            current_row = int(i[0])
            current_column = int(i[1])
            t_list = search_around(current_row, current_column, "T")

            for t in t_list:
                current_row = int(t[0])
                current_column = int(t[1])
                for row in range(-1, 2):
                    for col in range(-1, 2):
                        if (string_container[current_row+row][current_column+col] == "L"):
                            result_list.append(f"K{int(k[0])+1}{int(k[1])+1} M{int(m[0])+1}{int(m[1])+1} I{int(i[0])+1}{int(i[1])+1} T{int(t[0])+1}{int
                            (t[1])+1} L{current_row+row+1}{current_column+col+1}")

draw_table()

for item in result_list:
    print(item)
print(f"KMITL Count = {len(result_list)}")
```

```
ROW/COL   1    2    3    4    5
   1    ['*', '*', '*', '*', '*']
   2    ['*', 'M', 'M', '*', '*']
   3    ['*', 'K', 'I', 'K', '*']
   4    ['*', 'I', 'T', '*', '*']
   5    ['*', '*', 'L', '*', '*']


K32 M22 I33 T43 L53
K32 M23 I33 T43 L53
K34 M23 I33 T43 L53
KMITL Count = 3
```

```
ROW/COL   1    2    3    4    5
   1    ['*', '*', '*', '*', '*']
   2    ['*', 'M', 'M', '*', '*']
   3    ['*', '*', 'I', 'K', '*']
   4    ['*', 'I', 'T', '*', '*']
   5    ['*', '*', 'L', '*', '*']


K34 M23 I33 T43 L53
KMITL Count = 1
```