Design Doc

Overview:

• In this program, you will be able to choose minimum dimension in 3x3, and maximum 10x10 puzzle. For 3x3 puzzle, it has a square framed board consisting of 8 numbers from 1 to 8, with a blank space where an adjacent tile can slide to it. The object of this game is to rearrange the randomized puzzle into a sequential order by their number(left to right, top to bottom repeatedly. At the end of the game, the users can choose by themselves whether they want to restart a game of quit the game.

Data Model:

• how to represent the puzzle:

- A nested list:
  - [[1, 2, 3], [4, 5, 6], [7, 8, 0]] for a 3 dimension puzzle

• how to represent the represent key:

- allow user to input the character in the keyboard to represent each direction

• how to collect the steps trying to solve the puzzle:

- g_counter:

  Initially g_counter equals to 0, each time the user makes a move the g_counter will plus one, in the end, return g_counter as number of steps to move

• **Program Structure:**

- import random library to the program
  - the random will be used to randomized the sequence of direction in the main part of the program
- function definition:
  - def Initialize the puzzle boarding: sliding_puzzle_ini(dim):
    - initialize the ordered form of puzzle
  - def print the puzzle : puzzle_print(dim, puzzle_boarding):
    - print the puzzle
  - def get_blank_place(dim, puzzle_boarding):
    - return the row number and column number of blank

- def move(direction, puzzle_boarding, move_blank):
    - move the adjacent tile to target direction to the blank
- def available_movement(puzzle_boarding):
    - check the available move can be applied
- def check_success(puzzle_boarding):
    - check if the game has passed
- def key_press(keyword_press):
    - return a list that contain the available keyword represent each direction

- **processing Logic:**

main program:

- input the dimension of the puzzle to g_dim, report error until g_dim in range of 3 to 10, initialized the puzzle by using sliding_puzzle_ini(), to initialize a sliding puzzle game in the ordered form
- input the keywords to the g_keyword_press, use g_keyword_list to extract the first for alphabet character, correspondingly assign characters in g_keyword_list to left, right, up, down
- then assign [left, right, up, down, left, right, up, down, left, right, up, down] to the g_direction_list, randomized the puzzle, by move() function to move the direction contained in the g_direction_list. print the randomized puzzle
- g_counter is used to use record the number of steps used to soleve the game. (check_success(g_puzzle_boarding) == False) to examine the game does not go to the end.
- In the while loop available_movement() is used to analyze the available move for the adjacent tile of blank. After the user input the desired move, move() will move the target tile to the blank and return the outcome to g_puzzle_boarding. Print the puzzle by puzzle_print().
- after the while loop goes to Ture, the program will enable users to input n to start a new game and q to quit the game.

- **functional spec:**

- Initialize the puzzle boarding: sliding_puzzle_ini(dim):
    - the range of its parameter is from 3 to 10
    - double for loop to fill the number from 0 to dim**2 – 1 in the puzzle, 0 is to represent

blank place here.

- return the puzzle boarding (what it will be like when user passes the game)

- print the puzzle : puzzle_print(dim, puzzle_boarding):
  - replace the 0 in the puzzle with blank
  - use \t to make this whole boarding look better
  - print the puzzle on the screen

- get_blank_place(dim, puzzle_boarding):
  - using the double for loop to find the place of 0 in the puzzle
  - return the column number and row number of 0

- move(direction, puzzle_boarding, move_blank):
  - direction is the keyword to represent each direction (left right up down) in the main program
  - Move_blank is the list that contain the specific sequence of each direction the tile in puzzle will move to the blank
  - get_blank_place is used to find the place of blank
  - returns the puzzle boarding after moving

- available_movement(puzzle_boarding):
  - after the game has begun use get_blank_place to find the blank in the puzzle
  - for any position of the blank in the puzzle, to find the available direction the adjacent tile can move to
  - return the list of each available move

- check_success(puzzle_boarding):
  - use as a condition if the user can pass the game
  - counter is used to traverse the puzzle from the beginning of the puzzle to the end
  - return true if the counter goes to the end

- key_press(keyword_press):
    - The parameter is string user inputs in the main part
    - to examine whether the user input the correct form of keyword to represent each direction
    - use ascii to check the range of character represents alphabet character.

- Sample Output:

```
Microsoft Windows [版本 10.0.18363.1440]
(c) 2019 Microsoft Corporation。保留所有权利。

D:\csc1002>D:/python/python.exe d:/csc1002/A1_SDS_120040077_Source.py
Enter the desired dimension of the puzzle >3
Enter the four letters used for left, right, up and down directions >adws
1       2       3
4       6       8
7       5
s-down  d-right
enter your move >s
1       2       3
4       6
7       5       8
w-up  s-down  d-right
enter your move >d
1       2       3
4               6
7       5       8
w-up  s-down  a-left  d-right
enter your move >w
1       2       3
4       5       6
7               8
s-down  a-left  d-right
enter your move >a
1       2       3
4       5       6
7       8
congratulation! You solved puzzle in 4 step


Enter n to start a new game or enter q to end the game >
```

```
Enter n to start a new game or enter q to end the game >n
Enter the desired dimension of the puzzle >3
Enter the four letters used for left, right, up and down directions >adws
1       6       2
4               3
7       5       8
w-up  s-down  a-left  d-right
enter your move >s
1               2
4       6       3
7       5       8
w-up  a-left  d-right
enter your move >a
1       2
4       6       3
7       5       8
w-up  d-right
enter your move >w
1       2       3
4       6
7       5       8
w-up  s-down  d-right
enter your move >d
1       2       3
4               6
7       5       8
w-up  s-down  a-left  d-right
enter your move >w
1       2       3
4       5       6
7               8
s-down  a-left  d-right
enter your move >a
1       2       3
4       5       6
7       8
```

```
4               3
7       5       8
w-up  s-down  a-left  d-right
enter your move >s
1               2
4       6       3
7       5       8
w-up  a-left  d-right
enter your move >a
1       2
4       6       3
7       5       8
w-up  d-right
enter your move >w
1       2       3
4       6
7       5       8
w-up  s-down  d-right
enter your move >d
1       2       3
4               6
7       5       8
w-up  s-down  a-left  d-right
enter your move >w
1       2       3
4       5       6
7               8
s-down  a-left  d-right
enter your move >a
1       2       3
4       5       6
7       8
congratulation! You solved puzzle in 6 step


Enter n to start a new game or enter q to end the game >
```