Design Doc

Overview:

This program is a sliding puzzle. Its size can be designed by the users(from 3 ×3 to 10 ×10) . For a 3\*3 size, it has a square-framed board consisting of 8 square titles, numbered from 1 to 8, initially placed randomly. Users can also choose four different letters to represent four directions while controlling the movement. Every time they need to move the white board, they can get instruction that which direction they can move. Users win when the numbers appear sequentially, ordered from left to right, top to bottom. After that, they can choose whether to continue or not. Specially, the program users to keep entering letters or numbers if the currents do not satisfy the current standard.

Data model:

I create a single -dimensional list to store the numbers.

List can be changed and it is convenient to print. When finished, the numbers in the puzzle should be ordered and the white block is the last one. Thus, the largest number could be a good choice to represent it. Then we can compare the current list with the sorted one to decide whether users win. While playing, we can switch the number of white board and the target. The next time, just find where is the white block and repeat the same process. This process can be repeated until win.

Program Structure:

Design:

1. define a function to check whether the input number is valid and keep entering until it is valid
2. Use “while true” to let users plug in valid numbers
3. Create a list depending on the size of the puzzle
4. Make the puzzle solvable and display it

Display:

Define a function to display the puzzle. For the largest number, represent it with “ ”. Then, change another line when the amount of number in a line reach “size”.

Movement Control:

define a function to find the white block and switch it with the target number

Judgement:

1. use a “while” loop to control whether the uses win or not: if they win, the order of the list should be equal to the sorted list
2. define a function to start another game
3. add the another() function at the end to see whether to start another game

Processing Logic:

About the parts of my program:

1. for the steps that need to be repeat many times (e.g. movement control, display) create a function to make it simple
2. use a while loop to continue moving until the order is sorted
3. to avoid invalid input, set a “while loop” to control the letters input and use “while true” to control the numbers input
4. the whole prosses: first design the game (both size and movement) Then mess it randomly and display the beginning step. Third, keep playing by using a while loop and use +1 to count the steps. After finished, print the steps and use a function to simplify the other process.

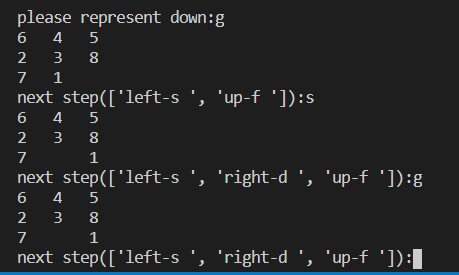
Initial puzzle:

At the beginning, I create a list that is in order. Then, I import the function “random” to select directions then move the blocks with the function “play()”. After repeating the process for 100 times (several may be invalid move), I get a solvable puzzle.

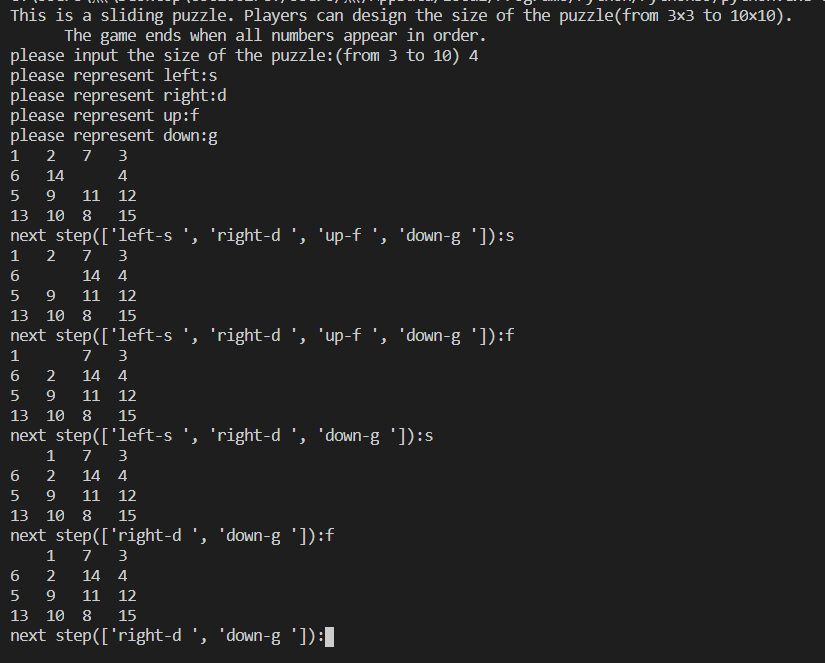
Functional Spec:

1. display(): This function is used to display the puzzle. To control when to change another line, I use “count” as a monitor. When count become the multiples, should change another line. And while printing, we can use ”%” to make the numbers aligned. I also change the largest number into “ “ as well.
2. Play() : This function is used to switch the numbers. At first, I use “index” to find the white block. Then, switch the number according to their position(index number). In some special position, we cannot move the block.
3. testIsL(): This function is used to test whether the letters are valid. They should not be the same and they should be single letters. If monitor called IsL is “False”, the users need to enter again.
4. PlayOnce(): This function is used to decide when to stop and count the steps. While the current list is not equal to the sorted one, the game should continue. And every time user move the block, count add one.
5. Another (): This function is used in the end. Users can choose to continue or stop. Also, we need to avoid invalid input.
6. validDirection(): This function is used to find the valid direction and let the users input next step. Valid directions are stored in a list.

Sample output:



(3×3)



(4×4)