Design Document

1. **Design:**

   a. **Overview:**

This program allows the user to play a sliding puzzle game of the numbers. The matrix of numbers ranges from 3 × 3 to 10 × 10. And it is fully random.

For the operation of the game, it supports custom operating keys by letters one by one. This is equipped with notice for invalid operation or input in game. In addition, the program will record and show the total steps at the end of one game. I also add the “once again” choice for game-lovers to challenge themselves.

   b. **Data model:**

The core object of the program is the matrix. To storage, show the matrix in order, I decide to choose “double-list” to achieve, which means the matrix is a list in fact and its items are lists as well, representing the items of a row. Then, we can use the first index to locate the row number and the second one to locate the column from 0. This is helpful to distinguish if the space is at the edge. For operation, I store custom keys in a list in the order of “up”, “down”, “left” and “right”, then I can use index in the followings.

   c. **Program Structure:**

To design the program, it is clear that it contains certain modules:

1. Input (the matrix size and operation keys)
2. Matrix storage
3. Operation and randomize
4. Matrix Display
5. Notice

When the game starts, input part allows the user to determine the size of the matrix and operation keys. And then according to the user, matrix storage initializes with requirements, generating the initial matrix in order. Then operation mode mess it randomly and convey it to display module. Then it is a loop for the input, (notice), operation, display, (notice). Details will be provided in the followings.

d. Processing logic:

1. Ask the user to decide size and operation keys (with input test).
2. Generate the origin matrix with requirement in order and copy it to a variable.
3. Randomize the matrix (by random() and operation()) (See mess() in function specification).
4. Display the matrix.
5. Ask the user to operate.
6. Validate the operation (with test, if false go 5).
7. Make the change to the matrix.
8. Compare with the saved one in 2 (if false go 5).
9. Once again? (if true, go 1).
10. End

2. Function specification:

integer_test(n): to test if the input n is an integer from 3 to 10, return Boolean value.

bind_key(): ask the user to input 4 keys for operation with a loop to test if it is a letter and user before. Return a list with four strings stands for “up”, “down”, “left”, “right” in order.

generate(n, total_List): to generate a matrix (total_List) with n $\times$ n in order and return. Define the location of the space with gl_Place.

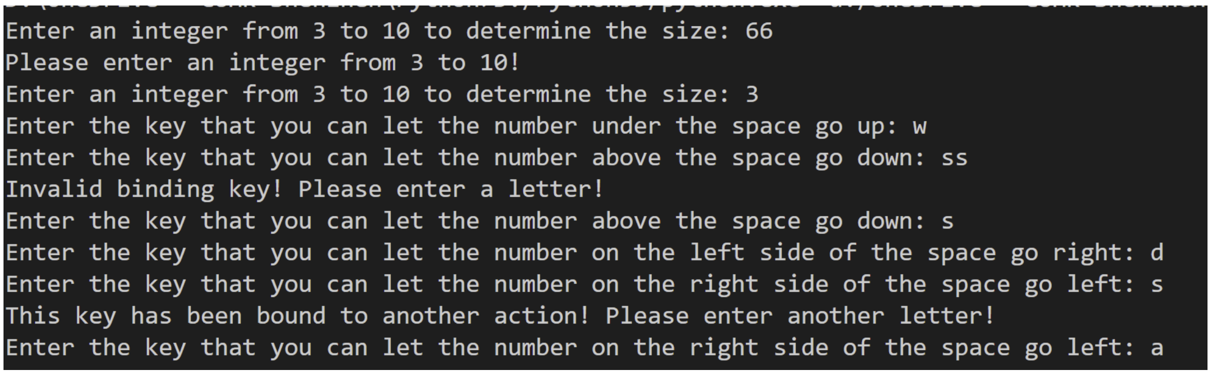
op_input(operation_List, gl_Place, n): to ask the user to operate and remind what operation is available according to operation_List, gl_Place and n. Return the input operation.

operation(op, l, operation_List, n): match the operation (string) with the index of operation_List to find the operation in fact. According to gl_Place and n, decide to whether to make the change to l (total_List). Return l (total_List).

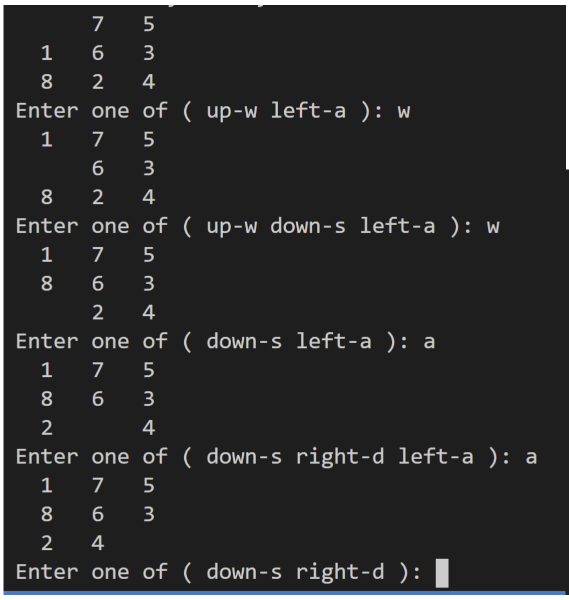
mess(a, operation_List, n): use random() to choose an item in operation_List randomly for $100n^2$ times and apply to a (total_List). Return a (total_List).

display(a): print a (matrix). "%+3s" % column to make sure that it is right aligned. Return a (matrix).

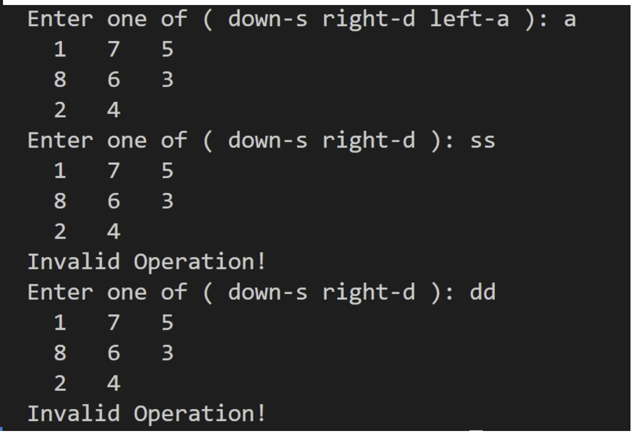
3. Sample output:



```
Enter an integer from 3 to 10 to determine the size: 66
Please enter an integer from 3 to 10!
Enter an integer from 3 to 10 to determine the size: 3
Enter the key that you can let the number under the space go up: w
Enter the key that you can let the number above the space go down: ss
Invalid binding key! Please enter a letter!
Enter the key that you can let the number above the space go down: s
Enter the key that you can let the number on the left side of the space go right: d
Enter the key that you can let the number on the right side of the space go left: s
This key has been bound to another action! Please enter another letter!
Enter the key that you can let the number on the right side of the space go left: a
```



```
       7   5
   1   6   3
   8   2   4
Enter one of ( up-w left-a ): w
   1   7   5
       6   3
   8   2   4
Enter one of ( up-w down-s left-a ): w
   1   7   5
   8   6   3
       2   4
Enter one of ( down-s left-a ): a
   1   7   5
   8   6   3
   2       4
Enter one of ( down-s right-d left-a ): a
   1   7   5
   8   6   3
   2   4
Enter one of ( down-s right-d ):
```



```
Enter one of ( down-s right-d left-a ): a
   1   7   5
   8   6   3
   2   4
Enter one of ( down-s right-d ): ss
   1   7   5
   8   6   3
   2   4
Invalid Operation!
Enter one of ( down-s right-d ): dd
   1   7   5
   8   6   3
   2   4
Invalid Operation!
```