Matlab Programming – Psych 114M
15 Puzzle

KEY CONCEPTS

Functions/subfunctions, nested loops, randomization, debugging.

ASSIGNMENT

The assignment is to write a program that allows a user to play the 15-puzzle as well as variants of the puzzle such as the 8-puzzle, 24-puzzle, etc. A description of the 15-puzzle can be found on Wikipedia: http://en.wikipedia.org/wiki/15_puzzle. In the 15 puzzle, there are 15 numbered square tiles with one tile missing. The missing tile allows the other tiles to be moved around. The game starts with a random ordering of the numbered tiles (see left panel below). The goal is to get the numbered tiles in order (see middle panel). On each turn, only a tile that neighbors the empty position can be moved into the empty slot.

Example of a random ordering at start       Goal configuration       Impossible configuration to solve

<table>
<thead>
<tr>
<th>9</th>
<th>2</th>
<th>12</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

One of the challenges of this assignment is think carefully about the representation of the missing tile. There are lots of ways to program this game. You might also consider starting just with the 15 puzzle, and then rewriting the program to solve the general n-puzzle. In my own attempt to program this puzzle, I used the following set of Matlab commands and functions: rng, randperm, reshape, while/end, break, find, abs, fprintf, for/end, sum, and if/then.

REQUIREMENTS

1) You write the program that allows a user to play the game. You do not have to write any AI engine that solves the game.
2) You do not have to create a graphical user interface for this assignment. You can simply use the command window to visualize the puzzle and ask for user input. See below for an example of how the screen can look like.
3) The user should first be asked the size $n$ of the puzzle they want to play. With $n=4$, they would play the standard game with $n^2-1 = 15$ numbered squares. However, you should program the game such that any $n>2$ are acceptable inputs. (NOTE: to simplify the program, you could first program the 15 puzzle and then later revisit this requirement for the general $n$-puzzle)
4) Each game should start with a random configuration of the numbered tiles that is different from the goal configuration.
5) At each iteration, the user is asked for their next move in terms of a number from 1...\(n^2-1\). The program needs to prompt the user for another input if the input corresponds to an illegal move (the proposed tile does not neighbor the empty tile) or the input is a not a number.

6) The program should display the number of correctly placed tiles at each iteration.

7) The program should stop when the solution is reached OR the user wants to stop the game (e.g., by not entering any move).

8) Your Matlab program could be written as a single .m file that has one main function and two local functions (within the same program). Look up the Matlab documentation for a description of local functions (also called subfunctions).

9) One local function should visualize the current state of the puzzle and the number of correctly placed tiles.

10) Another local function should calculate the number of correctly placed tiles.

11) It is likely that you will make lots of mistakes during the course of this assignment. The best way to find problems with your program is to debug the program. When you demonstrate your program (see grading below), you will be asked to also demonstrate the ability to debug the program.

EXTRA CREDIT

Some starting positions are impossible to solve. The figure above, right panel, shows one example. You can make sure that the configuration is possible to solve by initializing the puzzle with the correct solution and then going through a random sequence of moves to scramble the puzzle. Write your program with this random scrambler to guarantee the solvability of the puzzle.

SCREEN EXAMPLE

```matlab
>> FifteenPuzzle1
What is size of puzzle? 5
Current board: (#correct=1)
3    7   8  10   2
15  20   9  18  21
6  13   1  16
19   4  17  12  14
5  23  11  24  22

What is your move? 21
Current board: (#correct=1)
3    7   8  10   2
15  20   9  18
6  13   1  16  21
19   4  17  12  14
5  23  11  24  22

What is your move? 18
Current board: (#correct=1)
3    7   8  10   2
15  20   9  18
6  13   1  16  21
19   4  17  12  14
5  23  11  24  22

What is your move?
```

GRADING
This assignment is worth 100 points (before extra credit).

Feel free to ask the instructor and TA for feedback on preliminary versions of your program, especially when you get stuck on some aspect of the assignment. This feedback is provided only in class. Your grade for this assignment is based on three elements

1) The in-class demonstration of your program (for the TA and/or instructor)
2) Ability to answer basic questions (in class) about your code
3) Ability to use the debugger (in class). For example, the TA / instructor might make a slight modification to your program that leads to an error. You will have to demonstrate that you can step through the program and evaluate variables along the way, You should also be able to stop the program at the line where an error occurs.