Test cases

loadPuzzle()

<u>input validation</u>

- BufferedReader is null
- Number of slots (empty spaces in the puzzle) is less than n (number of words to solve)
- Number of slots (empty spaces in the puzzle) is greater than n
- First integer (Slot’s column) is greater than puzzle grid’s column
- Second integer (Slot’s row) is greater than puzzle grid’s row
- The letter at the end of part 2’s line contains something other than “h” or “v”
- Number of word lines is less than n
- Number of word lines is greater than n
- Blank line appears amid lines
- First line has fewer than three fields
- First line has negative integer for one it’s fields
- Slot line has fewer than four fields
- Slot line has negative integer in one of it’s fields
- Word lines contain duplicate words

<u>Boundary case</u>

- BufferedReader stream is empty
- row size is 0
- row size is 1
- row size is negative
- column size is 0
- column size is 1
- column size is negative
- number of words to solve is 0
- number of words to solve is 1

<u>Control flow</u>

- properly formatted stream with multiple words to solve
- puzzle with 0 connected slots.
- Puzzle with some connected slots.

<u>Data flow</u>

- Call with a different BufferedReader stream after already solving the puzzle with another stream.

solve()

<u>Input Validation</u>

- None

<u>Boundary case</u>

- number of words to solve is 0
- number of words to solve is 1
- puzzle with 0 connected slots.
- Puzzle with some connected slots.

<u>Control Flow</u>

- slot and word size does not match. (for a puzzle with two slots sized 3 and 4, the words are of size 3 and 5.)
- slot and word size matches.

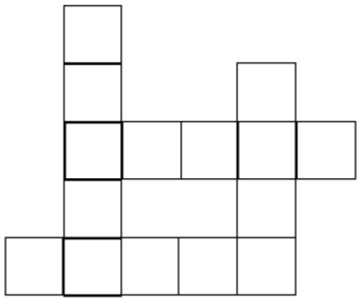

Figure 1 puzzle

- Given words will never be able to produce a solution(for example, for a puzzle like in figure 1, words would be bash, truck, frail, plush)
- Given words has the ability to produce a solution
- Words with unique length are given
- No word has unique length
- Only one solution exists for the puzzle

<u>Data flow</u>

- Call solve before loading a puzzle
- Call solve multiple times on a puzzle that can be solved in multiple ways

print()

<u>Input Validation</u>

- PrintWriter stream is null

<u>Boundary Case</u>

- None

<u>Control flow</u>

- None

<u>Data flow</u>

- Call print after puzzle is solved
- Call print before solving the puzzle
- Call print before loading the puzzle

choices()

Input Validations

- None

Boundary Case

- None

Control Flow

- Call choices for a puzzle that would not need backtracking to solve. An example of a puzzle like this might be, a puzzle with all distinct sized slots.
- Call choices for a puzzle that would need backtracking to solve.

Data Flow

- Call choices before loading a puzzle
- Call choices after loading but before solving a puzzle
- Call choices after solving a puzzle