Overview:

This program is about exploring state space search. This program is to fill in the puzzle in appropriate positions where a word could fit. The positions for these words can cross one another and two words are having common letter

A summary of the formatting requirements is in the CSCI 3901 course assignment #4 information in the course's bright space.

The Java program first reads the puzzle from the given stream of data. Depending on the words and places could fit in the puzzle.

Program flow:-

- LoadPuzzle reads a puzzle from the given stream of data. Created a grid structure with "#" placeholders and "*" cannot be filled with places. It returns true if puzzle is ready to read and solved. Returns false if some other error.
- Solve method returns true if the puzzle is solved and false if the puzzle cannot be solved with the given set of words. It solves the solution for given stream of input with the help of backtracking. By first fitting the words horizontally and then vertically.
- Print method prints the solution of the puzzle to the output stream and prints the actual solution puzzle.
- Choices method returns the number of guesses that your program had to make and later undoing while solving the puzzle.

Java files:

- FillInPuzzle.java – class file contains various operations to solve puzzle as load puzzle, solve and choices method

Modules :

- Read the input from the BufferReader stream.
- Parse each line in the stream for finding the size of puzzle and no of words to be fit into the puzzle.
- Figure the placeholders and fillable placeholders for words.
- Read each word and try to fit it in the horizontal places using backtracking.
- After fillings words horizontally and then try fitting words in vertical places using backtracking.
- After fitting check if all words are filled, then solution is found.

Assumptions:

- No word is repeated in the set of input words.
- The puzzle is case invariant.

- The words in the puzzle all fill from left to right or from top to bottom.

Data Structure and algorithm approach:

- First identify placeholders on the board. Try fitting each placeholders with each word that fits and if every placeholders can be filled without conflict solution is found.
- Pull one free placeholder from the grid.
- If all placeholders are empty or all placeholders are filled, stop solving the puzzle.
- Will check for the length of the word.
- For each word having exact length.
- If parts of the placeholders are filled, check for conflict.
- If the word does not fit into the placeholders, continue to next word from the input.
- Fill the placeholder with the word for the placeholder without conflict.
- Try next placeholder in down level.
- Start from the initial by pulling a free placeholder.
- If the recur found no solution and try next word.
- If no words fits in a grid, an above level need to try another word.

Steps followed for program efficiency:

- Grid is stored in a 2 dimensional character array for faster retrieval.
- After creating the grid, entire grid is scanned for slots in two passes from two directions.
- The two direction cells are filled using one single loop, differ mainly in how the slots are filled.
- Backtracking algorithm is used to reduce the time complexity and no of choices is displayed.