Project Report of Puzzle Solver
Shanshan He (sh2df), Lin Gong (lg5bt)

Introduction
In our project, we solve the puzzle problem with two different algorithms, brute force and dancing link. Also, we use two different shapes to represent our tiles, square and hexagon. By testing the given templates, dancing links turns out to be more effective than brute force search.

Implementation
As for the two algorithms, we have the same data structure to represent the tiles.

Brute Force
For brute force, we consider all the possible positions in the target board of each tile, including rotating and flipping it. We exhaustively search all the possible situations by putting down tiles one by one on the target board. The putting order is chosen according to the id of the tiles, we first choose tile 1 to fit to the board and then 2, and then 3, 4, ...n. Once tile i doesn't fit well, we go back to tile i-1 to try other possibilities. Since the search is exhaustive and the searching space is much larger, the algorithm turns out to be not very effective.

Dancing Links
Dancing link is the technique to effectively implement Algorithm X. It is heuristic recursive, non-deterministic, depth-first and backtracking, which is more effective than brute force. The algorithm transits the puzzle problem into an exact cover problem. We represent the board as an array by sticking the rows together. For each tile, we convert it into the same row format with the position of the unit on the target board. For the unit covered by this tile is set to 1, others are set to 0s. And we construct such a matrix with m columns and n rows where m is the size of target board and n is the total number of possible positions of all tiles. And our problem is reduced to finding a subset of rows which, when combined together, contain exact one 1 in each column. The step of searching algorithm is as follows:

1) Choose the column, which contains the lowest number of 1s. For those 1s in that column, we considered each row the 1 is at, lets say row {A, B, ...}.
2) For row A, we find all the 1s in row A, for those columns {i, j, k...} which contain those 1s in row A, we delete all the rows which contains the 1s in each column i, then delete the columns {i, j, k...}, the matrix is reduced to a smaller matrix. (The columns are removed because they have been filled, and the rows are removed because they conflict with the selected row.)
3) Repeat step 1) and 2) until the matrix is empty and the solutions is the rows we pick up each time. If the matrix isn’t empty but the lowest number of 1s in the columns is zero, then the branch of algorithm terminates unsuccessfully. Then the algorithm moves back one level and consider another row.

Results
Figure 1 is the screenshot of our application interface. We can choose the shape of the tiles and the algorithm to run the program.
After browsing the tile file, we can draw the tiles and target board. By clicking the button “show solution”, the program will find all the solutions in background. After it is done, it will show all the solutions on screen. By clicking “next solution” button, we can the next solution.

Figure 1 The interface of our program

Figure 2 The solution for IQCreator with square
As we can see, Figure 2 is the solution for “IQCreator” with the shape of square and Figure 3 is the solution for “checkerboard” with the shape of hexagon.