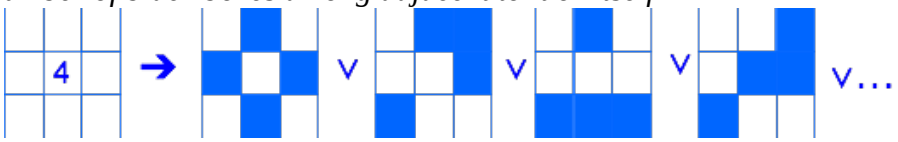
Name: TRAN, Chien Xuan ID: 1510032

**Report:** Implement a puzzle solver using miniSAT for the following problem. *Rule: the number in the box is the number of black boxes among adjacent to it or itself*




**Explanation**

Each cell *[i, j]* in the grid will be represented by logic variable *x[i, j]*. The main idea is:

- If there is a digit in a cell, we will get list of all variables representing adjacent boxes and itself. It is easy to see there are maximum 9 variables. In case the cell is at the corner or on the edge of the grid, then the number of variables will be less than 9. Denote this list as *N*.
- Generate all combinations *C* of *d* elements from above variables (*d* is the value of the cell)
- For each *C*, we have the following clause:

$$\left(\bigwedge_{x_k \in C} x_k\right) \wedge \left(\bigwedge_{x_l \in (N-C)} \neg x_l\right)$$

All these clauses are joined using OR operation (in DNF). Therefore, it is necessary to convert them into CNF. This is done using Tseitin conversion by introducing new variables *Example:*

```
(x1 & !x2 & !x3) | (!x1 & x2 & !x3)
```

will be converted to

```
(y1 | y2) & (!y1 | x1) & (!y1 | !x2) & (!y1 | !x3) & (!y2 | !x1) & (!y2 |
x2) & (!y2 | !x3)
```

**How the program works**

Here are the 4 steps performed during the execution of the program.

1. *Read input file*
2. *Generate CNF file*
3. *Run miniSAT against CNF file*
4. *Read miniSAT output file and save the human-readable result to file*

**Input file**

Use dot (.) character for an empty box. An example of valid input file:

```
. 2 3 . . 0 . . . .
. . . . 3 . 2 . . 6
. . 5 . 5 3 . 5 7 4
. 4 . 5 . 5 . 6 . 3
. . 4 . 5 . 6 . . 3
. . . 2 . 5 . . . .
4 . 1 . . . 1 1 . .
4 . 1 . . . 1 . 4 .
. . . . 6 . . . . 4
. 4 4 . . . . 4 . .
```

**MiniSAT path**
Please be noted that you have to update your miniSAT execution path to make sure the program will run properly. Update should be done on the constant *MINISAT_PATH* in *puzzle_solver.rb*

**How to run the program**
```
ruby puzzle_solver.rb <INPUT_FILE>
```

By default, the final output file will be named after the input file with suffix "_result".

**Development enviroment*
*+ Mac OSX Yosemite*
*+ Ruby 2.2.0*