<Team 8 Programming 1>

이름 : 우상흠, 김진수, 전영원

    Puzzle No.1 is a puzzle called 'kakurasu'. There is a grid of 8x8 and each cell is filled with either "o" or "x". There are integers written in the end of each row and column. The integer in the row represents the sum of the index of column with x. Similarly, the integer in the column represents the sum fo the index of row with x. We wrote a program in C that prints the puzzle that gets a file with sum written in a row and column.

    This program is largely divided into three modules. In first module, input.txt is inputted and stored in a two-dimensional array(a[ ][ ]). It calls the file input function 'fopen' and read the whole number of rows and columns in the array by 'fscanf'. At first, I didn't know why but I haven't read it well. After many trial and error, I could read one word in the type of character.

```
f = fopen("input.txt", "r");
```

```
fscanf(f,"%c ", &a[i][j]);
```

    The second module creates a formula that can be solved by z3 solver. After creating a formula.txt for z3 to read, each cell is declared as an integer. Then it fills each cell with one of 0, index of row, or column. The purpose is to use zero if not summed, index of row if used for sum of columns, and index of column for sum of rows.

```
fprintf(f2, "(declare-const a%d%d Int)\n", y, x);
```

=>

```
(declare-const a11 Int)
(declare-const a12 Int)
(declare-const a13 Int)
(declare-const a14 Int)
(declare-const a15 Int)
(declare-const a16 Int)
```

```
for (y = 1; y <= 8; y++) {
    for (x = 1; x <= 8; x++) {
        fprintf(f2, "(assert (or (= a%d%d 0) (= a%d%d %d) (= a%d%d %d)))\n", y, x, y, x, y, y, x, x);
    }
}
```

=>

```
(assert (or (= a11 0) (= a11 1) (= a11 1))
(assert (or (= a12 0) (= a12 1) (= a21 2))
(assert (or (= a13 0) (= a13 1) (= a31 3))
(assert (or (= a14 0) (= a14 1) (= a41 4))
(assert (or (= a15 0) (= a15 1) (= a51 5))
(assert (or (= a16 0) (= a16 1) (= a61 6))
```

And then, it uses 'assert' to specify the logic that the sum of a row or column equals the value stored in the two-dimensional array.

```
for (y = 1; y <= 8; y++) {
    fprintf(f2, "(assert(= (+");
    for (x = 1; x <= 8; x++) {
        fprintf(f2, " a%d%d", y, x);
    }
    fprintf(f2, ") %d)", a[0][y - 1]);
```

=>

```
(assert (= (+ a11 a12 a13 a14 a15 a16 a17 a18 ) 14))
(assert (= (+ a21 a22 a23 a24 a25 a26 a27 a28 ) 18))
(assert (= (+ a31 a32 a33 a34 a35 a36 a37 a38 ) 30))
(assert (= (+ a41 a42 a43 a44 a45 a46 a47 a48 ) 33))
(assert (= (+ a51 a52 a53 a54 a55 a56 a57 a58 ) 29))
```

Based on the problem resolved with z3, the last module specifies 'o' if the value is zero and 'x' if it is not zero. Though we thought the logic was correct and set the right command, z3 kept saying 'unsat'. We studied for a long time because the logic was a little difficult and the use of z3 was unfamilar, but unfortunately we could not solve the kakurasu puzzle. Using the first-time logic and programs was difficult and took a long time, but we learned a lot and realized the benefits of a computer.

Puzzle No.2 is a Puzzle called Sudoku. Sudoku is a puzzle in which the numbers in the horizontal, vertical, and 3x3 grids must be unique from 1 to 9.

This program is largely divided into three modules. This is file_open_and_generate_formula() and solve_formula() and interprete_Z3_output().

```
void file_open_and_generate_formula();
void solve_formula();
void interprete_Z3_output();
```

In first module, input.txt is inputted and stored in a two-dimensional array. It calls the file input function 'fopen' and read the whole number of rows and columns in the array by 'fscanf'. And then, it create formula.txt so that z3 can read and execute it.

In generate formula, there are many kind of variables that prevents the area's duplicated assertion.

```
//generate formula
int x, y;
int v = 0; //A variable that prevents the vertical assertion functions from being de-duplified.
int t = 0; //A variable that prevents the horizontal assertion functions from being de-duplified.
int r = 0; int s = 0; //3x3 part variable
int e = 2; //A variable that prevents the 3x3 area assertion functions from being de-duplified.
```

After the variable declaration, the code was implemented and entered into the file to open Formula.txt and solve the suduku according to the z3 grammar. The grammar format is this:

```
(declare-const a11 Int)
(declare-const a12 Int)
(declare-const a13 Int)
(declare-const a14 Int)
```
→ This is a label that integer value declaration text.

```
(assert (= a12 2))
(assert (= a14 5))
(assert (= a18 9))
(assert (= a21 8))
```

This is an assign a value that input.txt already given. →

```
(assert (and (<= a11 9) (<= 1 a11)))
(assert (and (<= a12 9) (<= 1 a12)))
(assert (and (<= a13 9) (<= 1 a13)))
```
→ This is an assertion that the value of each label must has 1 to 9.

```
(assert(distinct a11 a21 a31 a41 a51 a61 a71 a81 a91 ))
(assert(distinct a11 a12 a13 a14 a15 a16 a17 a18 a19 ))
(assert(distinct a11 a12 a13 a21 a22 a23 a31 a32 a33 ))
(assert(distinct a12 a22 a32 a42 a52 a62 a72 a82 a92 ))
```

→ This part was implemented on area so that the numbers in this area were unique from 1 to 9.

Second part is solve the formula. This part load the formula.txt and execute in z3 solver. We use the system function. Command line command is implemented to be invoked and executed automatically within the code without the need to execute z3 silver.

```
void solve_formula() {
    system("z3 formula.txt>solution.txt");
    printf("runs Z3 to solve formula.txt and then receives the result...\n");
}
```

And last part is printer the output.txt. This part can printed to output.txt so we must interpreted the

```
(define-fun a71 () Int
  1)
(define-fun a51 () Int
  0)
(define-fun a31 () Int
  1)
(define-fun a11 () Int
  0)
(define-fun a18 () Int
  1)
(define-fun a17 () Int
  1)
(define-fun a16 () Int
  0)
(define-fun a15 () Int
  1)
(define-fun a14 () Int
  0)
(define-fun a13 () Int
  1)
(define-fun a23 () Int
  0)
(define-fun a12 () Int
  0)
(define-fun a62 () Int
```

output2 - 메모장

```
 X O X O X O O
 O X O X O X O X
 X O O X O X X
  O X X O X O O
  X O O X X O O
 O X X O O X X
 X X O X X O O
  O O X O O X X
```

solution - 메모장

```
(define-fun a71 () Int
  1)
(define-fun a51 () Int
  0)
(define-fun a31 () Int
  1)
(define-fun a11 () Int
  0)
(define-fun a18 () Int
  1)
(define-fun a17 () Int
  1)
(define-fun a16 () Int
  0)
(define-fun a15 () Int
  1)
(define-fun a14 () Int
  0)
(define-fun a13 () Int
  1)
(define-fun a23 () Int
  0)
(define-fun a12 () Int
  0)
(define-fun a62 () Int
  1)
(define-fun a42 () Int
```

formula's output. The output is this:

Microsoft Visual Studio 디버그 콘솔

```
reads an input form input.txt...
generaties a corresponding formula as formula.txt...
runs Z3 to solve formula.txt and then receives the result...
interprets the Z3 output and print out the solution to output.txt...
print out the solution to output.txt...
6 2 4 5 7 1 3 9 8
8 7 1 2 9 3 4 5 6
9 3 5 4 6 8 1 7 2
7 8 9 1 4 5 6 2 3
5 4 3 7 2 6 8 1 9
1 6 2 3 8 9 7 4 5
4 9 7 6 3 2 5 8 1
2 5 6 8 1 4 9 3 7
3 1 8 9 5 7 2 6 4
```

The rule of this puzzle No. 3 called '3 in a row' tells that the number of 'o' or the number of 'x' is the same, and there is no three consecutive 'o' or 'x' in a row and a column

So we store the input data in a variable using the c language, and using the data, we generate output date(the puzzle of solution) by linking c language with z3

The first goal was to logically solve the puzzle using z3. because 3 in a row is a game represented by o or x,

It is difficult to solve by function, proposition logic and boolean written in z3. So by changing 'o' to 1 and 'x' to 0 we focused on applying the rules of the game to z3 let us decribe how to make code configuration and how c program models a puzzle, 3 in a row as a logical formular.

For example, look at the this code.

```
for (y = 1; y <= 8; y++)
    {fprintf(fl, "(assert (= 4 (+  ");
for (x = 1; x <= 8; x++)
fprintf(fl, " a%d%d", y, x);
fprintf(fl, " )))\n  ");
    }
```

using the 'for' syntax in c language and writing 'assert' in formular.txt, we make a row or column sum to be 4 for the number of 'o' to be equal to the number of 'x' in a row and column .

In this way, if we follow three consecutive rule, we also make the sum of the three consecutive cells in a line to be greater than 0 and less than 3(using z3 term ;assert' 'and',assert,<   ).

By doing this,... we generate the formular text file in c language to be usable to z3 like this

if z3 accept this formular file, z3 can interpret formular and solve the puzzel....so generate solution text file!!

Subsequently, Solution data can be used to generate output data that conforms to the puzzle format

description on how to build and execute your program

1.   read input.txt in c language (f = fopen("3inarowinput.txt", "r");)

2. generate formular.txt   (fl = fopen("formula2.txt", "w");)

3. using this code (system("z3 formula2.txt>solution2.txt");), generate solution.txt using z3 (fll = fopen("solution2.txt", "r");)

4. generate output.txt (fl = fopen("output2.txt", "w");)

📄 formula2 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
(assert (and (<= a85 1) (<= 0 a85)))
(assert (and (<= a86 1) (<= 0 a86)))
(assert (and (<= a87 1) (<= 0 a87)))
(assert (and (<= a88 1) (<= 0 a88)))
(assert (= 4 (+   a11 a12 a13 a14 a15 a16 a17 a18 )))
 (assert (= 4 (+   a21 a22 a23 a24 a25 a26 a27 a28 )))
 (assert (= 4 (+   a31 a32 a33 a34 a35 a36 a37 a38 )))
 (assert (= 4 (+   a41 a42 a43 a44 a45 a46 a47 a48 )))
 (assert (= 4 (+   a51 a52 a53 a54 a55 a56 a57 a58 )))
 (assert (= 4 (+   a61 a62 a63 a64 a65 a66 a67 a68 )))
 (assert (= 4 (+   a71 a72 a73 a74 a75 a76 a77 a78 )))
 (assert (= 4 (+   a81 a82 a83 a84 a85 a86 a87 a88 )))
 (assert (= 4 (+   a11 a21 a31 a41 a51 a61 a71 a81 )))
 (assert (= 4 (+   a12 a22 a32 a42 a52 a62 a72 a82 )))
 (assert (= 4 (+   a13 a23 a33 a43 a53 a63 a73 a83 )))
 (assert (= 4 (+   a14 a24 a34 a44 a54 a64 a74 a84 )))
 (assert (= 4 (+   a15 a25 a35 a45 a55 a65 a75 a85 )))
 (assert (= 4 (+   a16 a26 a36 a46 a56 a66 a76 a86 )))
 (assert (= 4 (+   a17 a27 a37 a47 a57 a67 a77 a87 )))
 (assert (= 4 (+   a18 a28 a38 a48 a58 a68 a78 a88 )))
 (assert(and (< (+ a31 (+ a21 a11)) 3) (> (+ a31(+ a21 a11)) 0)))
```

C:\Users\jung9\OneDrive\문서\3-in-a-row.exe

```
reads an input form input.txt...
? ? ? ? ? ? ? ?
o o ? ? x o ? ?
? ? ? o ? ? ? ?
x ? x ? ? x ? ?
? ? ? ? ? x ? ?
o ? x ? o ? ? ?
? x x ? x x ? ?
? ? ? ? ? ? ? ?
generaties a corresponding formula as formula.txt...
runs Z3 to solve formula.txt and then receives the result...
interprets the Z3 output and print out the solution to output.txt...
print out the solution to output.txt...
x x o x o x o o
o o x o x o o x
o x o x o x o x
x o x x o x o o
x x o o x x o o
o o x x o o x x
o x x o x x o o
x o o x o o x x

--------------------------------
Process exited after 0.2469 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . . ▯
```