Name: **TRAN, Chien Xuan** Student ID: ***1510032***

Report 1: Given two binary numbers (n-bits and m-bits wth n≧m). Describe their addition in CNF. Furthermore, implement a conversion program to CNF sugguested form, and confirm it result by SAT solvers (e.g., miniSAT).

EXPLANATION
Let's assume that two input binary numbers are x[0]x[1]x[2]...x[n] and y[0]y[1]...y[n]. For example:

```
X = 100110 => x[0] = 0, x[1] = 1, x[2] = 1, x[3] = 0, x[4] = 0, x[5] = 1
Y = 1010 => y[0] = 0, y[1] = 1, y[2] = 0, y[3] = 1
```

(Please be noted that x[0] is corresponding to the right most digit of X. It is the same for Y, Z, C)

If X has ***n*** digits and Y has ***m*** digits, then the result Z and carry C will respectively have at most ***n+1*** digits. So, in total we will use ***3n + m + 2*** variables for CNF encoding.

Auxiliary predicates:

```
x xor y = (x ^ ¬y) ˅ (¬x ^ y) = (x ˅ y) ^ (¬x ˅ ¬y)

x xor y xor z
 = (x ^ ¬y ^ ¬z) ˅ (¬x ^ y ^ ¬z) ˅ (¬x ^ ¬y ^ z) ˅ (x ^ y ^ z)
 = (x ˅ ¬y ˅ ¬z) ^ (¬x ˅ y ˅ ¬z) ^ (¬x ˅ ¬y ˅ z) ^ (x ˅ y ˅ z)

at_least_two(x, y, z)
 = (x ^ y) ˅ (x ^ z) ˅ (y ^ z)
 = (x ˅ y) ^ (x ˅ z) ^ (y ˅ z)
```

Following are the propositional rules for adding two binary numbers n-bits and m-bits (n >= m):

```
z[0] <=> x[0] xor y[0]
c[1] <=> x[0] ^ y[0]
c[i+1] <=> at_least_two(x[i], y[i], c[i]) (with 0 < i < m)
c[i+1] <=> x[i] ^ c[i] (with m <= i < n)
z[i] <=> x[i] xor y[i] xor c[i] (with 0 < i < m)
z[i] <=> x[i] xor c[i] (with m <= i < n)
z[n] <=> c[n]
```

The "<=>" expression will be converted to CNF form like this:

```
a <=> b → (¬a ˅ b) ^ (¬b ˅ a)
```

Because we can not use character like *x, y, z* in the config file for SAT solver, we have to map all those variables into a standard list of variables in which each of them will be assigned a number starting from ***1*** to ***(3n+m+2)*** as follows:

1 → (n+1): Z
(n+2) → (2n+1): X
(2n+2) → (2n+m+1): Y
(2n+m+2) → (3n+m+2): C

Example:
Let's say we have generated the CNF coding for adding two number *1001101* (n=7) and *10101* (m=5), and run the SAT solver with the generated config file, and obtained the following file
SAT
-1 2 -3 -4 -5 6 7 -8 9 -10 11 12 -13 -14 15 16 -17 18 -19 20 -21 22 -23 24 25 26 -27 -28 0

Interpretation of this file is as follows. The first 8 variables represent Z:
z[0] = 0, z[1] = 1, z[2] = 0, z[3] = 0, z[4] = 0, z[5] = 1, z[6] = 1, z[7] = 0

so Z is 1100010 (again, notice that z[0] represents right most bit of Z)

How to run conversion program

The program is used to convert the addition of two binary numbers into DIMACS format which can be understood by SAT Solver (miniSAT in this case). It is written in Ruby (version 2.2.0). If Ruby has been installed in the system, run the following command

```
ruby addition.rb <INPUT_FILE>
```

The input file should contains a single line with two binary numbers separated by a white space. The MiniSAT configuration file will be generated in the same folder (by default, configuration file will be named ***"config.txt"***). We can then run ***miniSAT*** against this file to confirm the result.