Lab1 - Ex2

The python script `sudoku_conv` serves the purpose of transforming a Sudoku puzzle into a DIMACS file. This file contains clauses composed of:

- The initial clues derived from the Sudoku input file
- The constraints detailed in the research paper titled "Sudoku as a SAT Problem" by Lynce, I., and Ouaknine.

A SAT solver is supplied with this DIMACS file to solve the sudoku.

Python script: `sudoku_conv`

```
# Returns the dimacs CNF variable number for variable s{x,y,z}
#    • s{x,y,z} = at cell (x,y) we have the value v
def var(x, y, z):
    return 81 * (x - 1) + 9 * (y - 1) + z


# Converts a file that contains a sudoku puzzle to a dimacs format
def convert(fileName, outFileName):
    # Get the initial clues from puzzle
    clues = []
    with open(fileName, "r") as f:
        for line in f.readlines():
            clues.append(line.strip())

    # Build the clauses based on the constraints & initial clues
    # The constraints are takes from "Lynce, I., and Ouaknine, J. Sudoku as a SAT Problem"
    cls = []  # will hold lists of ints

    # Initial clues from the puzzle
    for x in range(1, 10):  # go from 1 to 10 because range will stop at end-1=9
        for y in range(1, 10):
            if clues[x-1][y-1] != '.':
                cls.append([var(x,y,int(clues[x-1][y-1]))])

    # C1. There is at least one number in each entry
    for x in range(1, 10):
        for y in range(1, 10):
            cls.append([var(x,y,z) for z in range(1, 10)])

    # C2. Each number appears at most once in each row
    for y in range(1, 10):
        for z in range(1, 10):
            for x in range(1, 9):
                for i in range(x+1, 10):
                    cls.append([-var(x,y,z), -var(i,y,z)])
```

```
    # C3. Each number appears at most once in each column
    for x in range(1, 10):
        for z in range(1, 10):
            for y in range(1, 9):
                for i in range(y+1, 10):
                    cls.append([-var(x, y, z), -var(x, i, z)])

    # C4. Each number appears at most once in each 3x3 sub-grid
    for z in range(1, 10):
        for i in range(0, 3):
            for j in range(0, 3):
                for x in range(1, 4):
                    for y in range(1, 4):
                        for k in range(y+1, 4):
                            cls.append([-var(3*i+x, 3*j+y, z), -var(3*i+x, 3*j+k, z)])

    for z in range(1, 10):
        for i in range(0, 3):
            for j in range(0, 3):
                for x in range(1, 4):
                    for y in range(1, 4):
                        for k in range(x+1, 4):
                            for l in range(1, 4):
                                cls.append([-var(3*i+x, 3*j+y, z), -var(3*i+k, 3*j+l, z)])

    with open(outFileName, 'w') as f:
        # Write the header to the file
        f.write("p cnf %d %d\n" % (729, len(cls)))

        # Write the clauses to the file
        for c in cls:
            f.write(" ".join([str(l) for l in c]) + " 0\n")


convert("test.txt", "out.txt")
```

Example for the input file containing sudoku:

```
.......1.
4........
.2.......
....5.4.7
..8...3..
..1.9....
3..4..2..
.5.1.....
...8.6...
```

The outcome obtained after running CaDiCaL with the DIMACS file generated by the Python script:

```
c --- [ result ] ------------------------------------------------------------
c
s SATISFIABLE
v -1 -2 -3 -4 -5 6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -17 18 -19 -20 21 -22
v -23 -24 -25 -26 -27 -28 -29 -30 -31 -32 -33 34 -35 -36 -37 -38 -39 -40 -41
v -42 -43 44 -45 -46 -47 -48 49 -50 -51 -52 -53 -54 -55 -56 -57 -58 59 -60 -61
v -62 -63 64 -65 -66 -67 -68 -69 -70 -71 -72 -73 74 -75 -76 -77 -78 -79 -80
v -81 -82 -83 -84 85 -86 -87 -88 -89 -90 -91 -92 -93 -94 -95 -96 -97 98 -99
v -100 -101 -102 -103 -104 -105 106 -107 -108 -109 -110 -111 -112 113 -114
v -115 -116 -117 118 -119 -120 -121 -122 -123 -124 -125 -126 -127 128 -129
v -130 -131 -132 -133 -134 -135 -136 -137 -138 -139 -140 -141 -142 -143 144
v -145 -146 147 -148 -149 -150 -151 -152 -153 -154 -155 -156 -157 -158 159
v -160 -161 -162 163 -164 -165 -166 -167 -168 -169 -170 -171 -172 173 -174
v -175 -176 -177 -178 -179 -180 -181 -182 -183 -184 185 -186 -187 -188 -189
v -190 -191 -192 -193 -194 -195 -196 -197 198 -199 -200 -201 -202 -203 204
v -205 -206 -207 -208 -209 210 -211 -212 -213 -214 -215 -216 -217 -218 -219
v -220 -221 -222 -223 224 -225 -226 -227 -228 -229 -230 -231 232 -233 -234
v -235 -236 -237 238 -239 -240 -241 -242 -243 -244 -245 -246 -247 -248 -249
v -250 -251 252 -253 -254 255 -256 -257 -258 -259 -260 -261 -262 263 -264 -265
v -266 -267 -268 -269 -270 -271 -272 -273 -274 -275 276 -277 -278 -279 -280
v -281 -282 -283 284 -285 -286 -287 -288 289 -290 -291 -292 -293 -294 -295
v -296 -297 -298 -299 -300 301 -302 -303 -304 -305 -306 -307 -308 -309 -310
v -311 -312 -313 314 -315 -316 -317 -318 -319 -320 -321 322 -323 -324 -325
v -326 -327 -328 329 -330 -331 -332 -333 -334 -335 -336 -337 -338 339 -340
v -341 -342 -343 -344 -345 -346 -347 -348 -349 350 -351 -352 353 -354 -355
v -356 -357 -358 -359 -360 -361 -362 -363 364 -365 -366 -367 -368 -369 -370
v -371 -372 -373 -374 -375 376 -377 -378 -379 -380 381 -382 -383 -384 -385
v -386 -387 -388 -389 -390 -391 -392 -393 -394 -395 396 397 -398 -399 -400
v -401 -402 -403 -404 -405 -406 -407 -408 -409 -410 -411 412 -413 -414 -415
v -416 -417 418 -419 -420 -421 -422 -423 424 -425 -426 -427 -428 -429 -430
v -431 -432 -433 -434 435 -436 -437 -438 -439 -440 -441 -442 -443 -444 -445
v -446 -447 -448 -449 450 -451 -452 -453 -454 -455 -456 -457 458 -459 -460
v -461 -462 -463 -464 465 -466 -467 -468 -469 470 -471 -472 -473 -474 -475
v -476 -477 -478 -479 -480 -481 482 -483 -484 -485 -486 -487 -488 489 -490
v -491 -492 -493 -494 -495 496 -497 -498 -499 -500 -501 -502 -503 -504 -505
v -506 -507 -508 -509 -510 -511 -512 513 -514 -515 -516 517 -518 -519 -520
v -521 -522 -523 -524 -525 -526 -527 -528 529 -530 -531 -532 -533 -534 -535
v 536 -537 -538 -539 -540 -541 542 -543 -544 -545 -546 -547 -548 -549 -550
v -551 -552 -553 -554 555 -556 -557 -558 -559 -560 -561 -562 -563 -564 -565
v 566 -567 -568 -569 -570 -571 -572 -573 -574 575 -576 -577 -578 -579 -580 581
v -582 -583 -584 -585 -586 -587 -588 -589 -590 591 -592 -593 -594 595 -596
v -597 -598 -599 -600 -601 -602 -603 -604 605 -606 -607 -608 -609 -610 -611
v -612 -613 -614 -615 -616 -617 -618 -619 -620 621 -622 -623 -624 -625 -626
v -627 628 -629 -630 -631 -632 -633 634 -635 -636 -637 -638 -639 -640 -641 642
v -643 -644 -645 -646 -647 -648 -649 650 -651 -652 -653 -654 -655 -656 -657
v -658 -659 -660 -661 -662 -663 664 -665 -666 -667 -668 -669 670 -671 -672
v -673 -674 -675 -676 -677 -678 -679 -680 -681 -682 683 -684 -685 -686 687
v -688 -689 -690 -691 -692 -693 -694 -695 -696 -697 -698 699 -700 -701 -702
v 703 -704 -705 -706 -707 -708 -709 -710 -711 -712 -713 -714 -715 716 -717
v -718 -719 -720 -721 -722 -723 -724 -725 -726 -727 -728 729 0
```