MAT 160 PROJECT 2

1. Write an integer programming model for the following problems (HINT: Think of the problem we discuss of making a party for people that may hate each other).

   (a) The city of SIVAD in the state of AINROFILAC has a problem with very predictable criminals. The city has n criminals and when k of the criminals all know each other (pairwise acquaintances) they try to form a gang. Assuming that the police knows each pair-wise friendship for each of the possible pairs of criminals (say these are given e.g., by an undirected network whose nodes represent criminals, edges represent criminals that know each other), how can the police decide the maximum possible size that a gang in town?

   (b) The police has decided to pay some informants from among the members of the criminal group. If each bad guy can only report to the police about those criminals he/she is acquainted with, write an integer program that can help the police compute the minimum number of criminals they need to be employed as informants.

a) $a_{ij} = \begin{cases} 1 & \text{if the ith criminal knows the jth criminal} \\ 0 & \text{otherwise.} \end{cases}$

$X_j = \begin{cases} 1 & \text{if the jth criminal doesn't know any criminal} \\ 0 & \text{otherwise.} \end{cases}$

Minimize criminals than doesn't know any criminal is the same to maximize the number criminals that a gang could have.

Objective: Minimize $\sum_{j=1}^{n} X_j$

Constant: Subject to: $X_i + X_j \leq 1$ for each a_{ij}

Since the ith criminal knows jth, then $x_i + x_j \leq 1$

b) $Y_i = \begin{cases} 1 & \text{if the ith criminal is employed} \\ 0 & \text{otherwise.} \end{cases}$

Minimize number of criminal employed

Objective: Minimize $\sum_{i=1}^{n} Y_i$

Constraint: Subject to $Y_i + Y_j = 2$ for each a_{ij}

Since $\boxed{\text{i knows j}}$, Y_i could be employed but also Y_j could be employed, so for this constraint the optimals Y_i will be the ones that know more people.

2. A vertex-cover of a graph G is a set S of vertices of G such that each edge of G is incident with at least one vertex of S. Formulate a discrete model that given a graph finds a vertex cover with smallest number of vertices. Explain the reasoning on your variables and constraints. Show that if M is a matching and S is some vertex cover $|S| \geq |M|$. In particular show that

$$\max\{|M| : M \text{ is a matching of } G\} \leq \min\{|S| : S \text{ is a vertex-cover of } G\}.$$

HINT: How many vertices do you need to cover just the edges of M?

Vertex Cover.

$X_i = \begin{cases} 1 & \text{if the ith vertex is chosen} \\ 0 & \text{otherwise} \end{cases}$

Minimaze $\sum_i X_i$

Subject to

$X_i + X_j \geq 1$ for each edge (i,j)

$0 \leq X_i \leq 1$

In this case, we pick an arbitrary edge. Since any vertex cover must hit at least 1 endpoint of it, take both endpoints. then, trow the out all edges covered and repeat the process. At the end, there must be no uncovered edges left

If M is a matching and S some vertex cover $|S| \geq |M|$

Proof: Let M be a maximum matching of G, and let S be a minimum vertex cover. Then, fore every edge $(i,j) \in M$, there exists and end pond of the edge (i,j) in S. thus, since edges of M share no ending points, $\boxed{|M| \leq |S|}$.

In other words, $\max\{|M| : M \text{ is a matching of } G\} \leq \min\{|S| : S \text{ is vertex cover of } G\}$

3. Give an example of an Integer Linear program which has no feasible integer solutions, but its LP relaxation has a feasible set in R^2 of area at least 100.

3.- Give an example of an Integer linear program which has no feasible integer solutions but its LP relaxion has a feasible set in $\mathbb{R}^2$ of area at least 100.

Maximize 0

Subject to $x_2 \geq \frac{1}{4}$

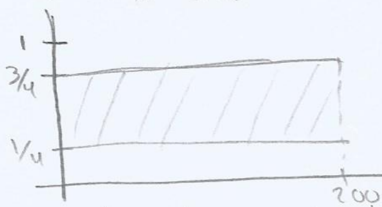
$x_2 \leq 3/4$

$x_1 \geq 0$

$x_1 \leq 200$

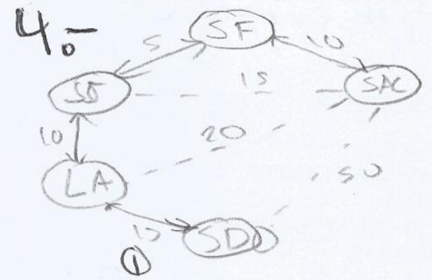
$x_1, x_2 \in \mathbb{Z}$

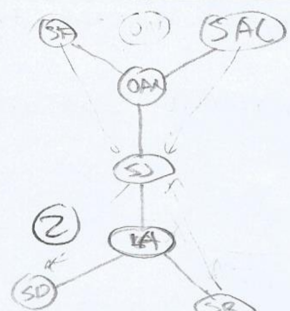
The integer linear program is infeasible but its LP relaxation has a feasible region that forms the area of 100.



4. Given a graph $G = (V, E)$ representing say the cities of California connected by highways, we wish to find out whether there is a tour of the vertices of G (a cycle visiting each vertex exactly once) which only uses the existing edges in E. Suppose you have a powerful software that solves the Traveling salesman problem. How would you use that algorithm for the TSP to answer that question? What costs should you pick on arcs?

4.-





In this case, we want to find out if there is a tour of the vertices of G (a cycle visiting each vertex exactly once) which only uses the existen edges in E.

As shown in class, for the graph 2, it is impossible to find a ~~path~~ such that each vertex is visited exactly once. Because no mather where you started you will be visiting the city of Oakland, for example at least twice. Now if we have an ~~program~~ algorithm that satisfies the TSP, we can use it to find the shortest and cheapest path. In this case, we ~~will~~ know for sure that the ~~cost~~ boundary will be ~~less~~ than equal to the ~~sum~~ of all values of the edges. In other words,

$\exists$ ~~K~~ $M = \sum_{i=1}^{\text{total edges}} C_{\text{edge } i}$ then $\exists K \in \mathbb{Z}$ s.t. $1 \leq K \leq M$

So the program will find the smallest possible value for K ~~which~~ which will give ~~you~~ us the optimal cost and path that we should take.

5. **Modeling an Advertising problem.** A company wants to promote its newly developed product by launching an advertising campaign. There are four advertising options to choose from: TV Spot, Newspaper, Radio (prime time), and Radio (afternoon); these options are labelled T, N, P, A respectively. The table below provides, for each type of advertising, the audience reached, the cost and maximum number of ads per week. The company has a budget of 8000 dolars per week and seeks to maximize audience reached. However, the company also wants 5 or more radio spots per week and cannot spend more than 1800 dollars on radio per week. Let T, N, P, A be the decision variables corresponding to the numbers of ads chosen weekly by the company. Formulate this as a linear integer programming problem in SCIP making sure to incorporate all the constraints in the formulation! Solve the problem using SCIP.

Advertising options	TV Spot (T)	Newspaper (N)	Radio (P) (prime time)	Radio (A) (afternoon)
Audience Reached (per ad)	5000	8500	2400	2800
Cost (per ad)	$ 800	$ 925	$ 290	$ 380
Max Ads (per week)	12	5	25	20

```
var T integer;
var N integer;
var P integer;
var A integer;

maximize profit:
  5000 * T + 8500 * N + 2400 * P + 2800 * A;

subto radio_constraint:
  290 * P + 380 * A <= 1800;

subto budget_constraint:
  800 * T + 925 * N + 290 * P + 380 * A <= 8000;

subto radio_size_constraint:
  P + A >= 5;

subto tv_constraint:
  T <= 12;

subto NP_constraint:
  N <= 5;

subto RP_constraint:
  P <= 25;

subto RA_constraint:
  A <= 20;
```

```
SCIP> read Advertisingproblem.zpl

read problem <Advertisingproblem.zpl>
============


base directory for ZIMPL parsing: </homes/home04/class/m160s1-6>

original problem has 4 variables (0 bin, 4 int, 0 impl, 0 cont) and 7 constraints
SCIP> optimize

presolving:
(round 1, fast)       0 del vars, 4 del conss, 0 add conss, 6 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd conss, 0 impls, 0 clqs
(round 2, fast)       0 del vars, 4 del conss, 0 add conss, 9 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd conss, 0 impls, 0 clqs
(round 3, exhaustive) 0 del vars, 4 del conss, 0 add conss, 9 chg bounds, 0 chg sides, 0 chg coeffs, 2 upgd conss, 0 impls, 0 clqs
presolving (4 rounds: 4 fast, 2 medium, 2 exhaustive):
 0 deleted vars, 4 deleted constraints, 0 added constraints, 9 tightened bounds, 0 added holes, 0 changed sides, 0 changed coefficients
 0 implications, 0 cliques
presolved problem has 4 variables (0 bin, 4 int, 0 impl, 0 cont) and 3 constraints
      2 constraints of type <varbound>
      1 constraints of type <linear>
transformed objective value is always integral (scale: 100)
Presolving Time: 0.00

 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|  dualbound   | primalbound  |  gap
T 0.0s|     1 |     0 |     0 |     - | 204k|   0 |   - |   4 |   3 |   4 |   3 |   0 |   0 |   0 |      --      | 1.200000e+04 |    Inf
b 0.0s|     1 |     0 |     0 |     - | 203k|   0 |   - |   4 |   3 |   4 |   3 |   0 |   0 |   0 |      --      | 6.690000e+04 |    Inf
  0.0s|     1 |     0 |     2 |     - | 203k|   0 |   2 |   4 |   3 |   4 |   3 |   0 |   0 |   0 | 6.718586e+04 | 6.690000e+04 |   0.43%
  0.0s|     1 |     0 |     2 |     - | 203k|   0 |   - |   4 |   3 |   4 |   3 |   0 |   0 |   0 | 6.690000e+04 | 6.690000e+04 |   0.00%

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.00
Solving Nodes      : 1
Primal Bound       : +6.69000000000000e+04 (2 solutions)
Dual Bound         : +6.69000000000000e+04
Gap                : 0.00 %

SCIP> display solution

objective value:                                66900
T                                                   2   (obj:5000)
N                                                   5   (obj:8500)
P                                                   6   (obj:2400)
```

6. **Computer PROJECT 1:** *Directed graphs are important for deciding order of activities:* A directed graph $G = (V, A)$ can be used to represent the order of actions to be take in a project (task a needs to be done *before* b, if there is an arc from a to b. A topological ordering of the vertices is assignment of the value y_i to each vertex such that for every arc ij then $y_i \geq y_j + 1$.

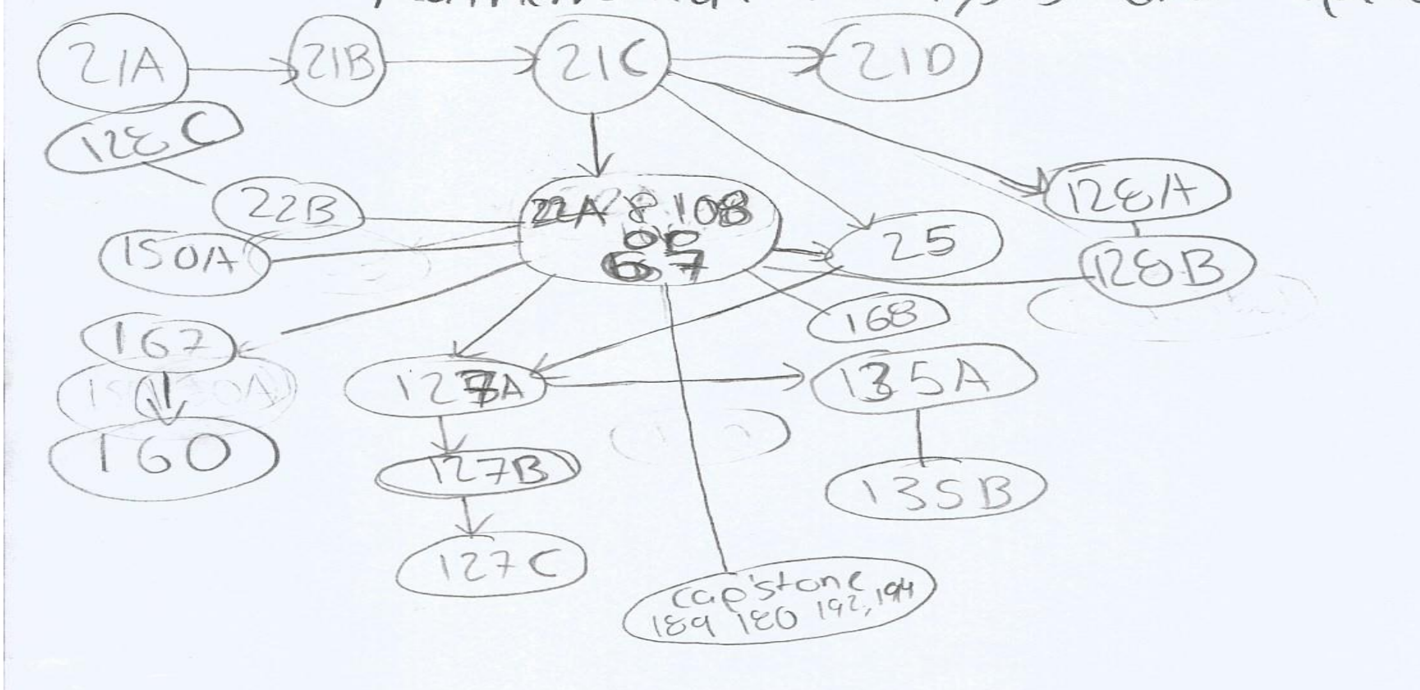
   (a) Show that a directed graph has a topological ordering, then there is no directed cycle. Write a simple discrete model to detect whether a directed graph has a cycle.

   (b) A UC Davis student in major X (say Applied Math) has to take certain courses that have pre-requisites. Create a directed graph whose nodes are all possible Math courses in each of the majors and there is an arc from course a to course b if course a is a pre-requisite to b. How big is this graph? Let's call this the Math-course graph $MathC$

   (c) Write a SCIP model that, given any of the 3 majors of the UC Davis mathematics department, tells the students at least one good order, one that does not skip pre-requisites, in which to take their math courses and graduate in minimum amount of time. Can you find at least two good orders in each of the majors? Make some comments about the structure of the $MathC$ graph.

6.- Computer Project 1.

Mathematical Analysis and Opera




```
# Number of classes

set C := {"21A","21B","21C","21D", "22A&108or67","25","22B","128A","135A","135B","127A","127B","127C", "160", "168", "167", "150A", "CPSN"};

set E := {<"21A","21B">, <"21B","21C">, <"21C","21D">, <"21C","22A&108or67">, <"21C","25">, <"21C","128A">, <"22A&108or67", "22B">, <"22A&108or67", "150A">, <"22A
&108or67", "167">, <"22A&108or67", "127A">, <"22A&108or67", "168">, <"22A&108or67", "135A">, <"22A&108or67", "CPSN">, <"167","160">, <"127A","127B">, <"127B","127
C">, <"135A","135B">, <"127B","CPSN">, <"21D","127B">, <"22B","167">};

set Q := { 1..12 };
set CQ := Q * C;

# Edge variable
var x[CQ] binary;
var z[E] binary;
var y[C] integer;

# Quaters
param quater[Q * C] :=
     |"21A","21B","21C","21D", "22A&108or67","25","22B","128A","135A","135B","127A","127B","127C", "160", "168", "167", "150A", "CPSN"|
  | 1|    1,    1,    1,    1,              1,   1,    1,     1,     1,     1,     1,     1,     1,     1,     1,     1,     1,      1|
  | 2|    2,    2,    2,    2,              2,   2,    2,     2,     2,     2,     2,     2,     2,     2,     2,     2,     2,      2|
  | 3|    3,    3,    3,    3,              3,   3,    3,     3,     3,     3,     3,     3,     3,     3,     3,     3,     3,      3|
  | 4|    4,    4,    4,    4,              4,   4,    4,     4,     4,     4,     4,     4,     4,     4,     4,     4,     4,      4|
  | 5|    5,    5,    5,    5,              5,   5,    5,     5,     5,     5,     5,     5,     5,     5,     5,     5,     5,      5|
  | 6|    6,    6,    6,    6,              6,   6,    6,     6,     6,     6,     6,     6,     6,     6,     6,     6,     6,      6|
  | 7|    7,    7,    7,    7,              7,   7,    7,     7,     7,     7,     7,     7,     7,     7,     7,     7,     7,      7|
  | 8|    8,    8,    8,    8,              8,   8,    8,     8,     8,     8,     8,     8,     8,     8,     8,     8,     8,      8|
  | 9|    9,    9,    9,    9,              9,   9,    9,     9,     9,     9,     9,     9,     9,     9,     9,     9,     9,      9|
  |10|   10,   10,   10,   10,             10,  10,   10,    10,    10,    10,    10,    10,    10,    10,    10,    10,    10,     10|
  |11|   11,   11,   11,   11,             11,  11,   11,    11,    11,    11,    11,    11,    11,    11,    11,    11,    11,     11|
  |12|   12,   12,   12,   12,             12,  12,   12,    12,    12,    12,    12,    12,    12,    12,    12,    12,    12,     12|;


# Minimize quaters
minimize major_quaters:
  sum <q,c> in CQ : quater[q,c] * x[q,c];

#Max/min classes each quater
subto max_classes:
  forall <q> in Q do
     sum <c> in C : x[q,c] <= 3;


subto min_classes:
  forall <q> in Q do
     sum <c> in C : x[q,c] >= 0;

#Take prerequisites first
subto oneafter:
  forall <c> in C do
    sum <q> in Q : x[q,c] == 1;

subto firstclasses:
  forall <i,j> in E do
    sum <q> in Q : quater[q,j] * x[q,j] - sum <q> in Q : quater[q,i] * x[q,i] >= 1;

~
```

```
 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|  dualbound   | primalbound  |  gap
T 0.1s|     1 |     0 |     0 |     - |1269k|   0 |   - | 102 |  53 | 102 |  45 |   0 |   8 |   0 |      --      | 1.480000e+02 |   Inf
  0.1s|     1 |     0 |    43 |     - |1261k|   0 |  24 | 102 |  53 | 102 |  45 |   0 |   8 |   0 | 9.600000e+01 | 1.480000e+02 | 54.17%
s 0.1s|     1 |     0 |    43 |     - |1266k|   0 |  24 | 102 |  53 | 102 |  45 |   0 |   8 |   0 | 9.600000e+01 | 9.900000e+01 |  3.13%
  0.1s|     1 |     0 |    65 |     - |1408k|   0 |  25 | 102 |  53 | 102 |  63 |  18 |   8 |   0 | 9.600000e+01 | 9.900000e+01 |  3.12%
  0.1s|     1 |     0 |    65 |     - |1407k|   0 |  25 | 102 |  53 | 102 |  54 |  18 |   8 |   0 | 9.600000e+01 | 9.900000e+01 |  3.12%
  0.1s|     1 |     0 |    77 |     - |1580k|   0 |  12 | 102 |  46 | 102 |  71 |  35 |   8 |   0 | 9.600000e+01 | 9.900000e+01 |  3.12%
  0.1s|     1 |     0 |    79 |     - |1656k|   0 |   0 | 102 |  46 | 102 |  72 |  36 |   8 |   0 | 9.600000e+01 | 9.900000e+01 |  3.12%
* 0.1s|     1 |     0 |    79 |     - |1658k|   0 |   - | 102 |  46 | 102 |  72 |  36 |   8 |   0 | 9.600000e+01 | 9.600000e+01 |  0.00%

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.12
Solving Nodes      : 1
Primal Bound       : +9.60000000000000e+01 (5 solutions)
Dual Bound         : +9.60000000000000e+01
Gap                : 0.00 %

SCIP> display solution

objective value:                                   96
x#1$21A                                             1   (obj:1)
x#2$21B                                             1   (obj:2)
x#3$21C                                             1   (obj:3)
x#4$22A&108or67                                     1   (obj:4)
x#4$25                                              1   (obj:4)
x#4$128A                                            1   (obj:4)
x#5$21D                             0.999999999999999   (obj:5)
x#5$22B                                             1   (obj:5)
x#5$127A                                            1   (obj:5)
x#6$127B                            0.999999999999998   (obj:6)
x#6$168                                             1   (obj:6)
x#6$167                                             1   (obj:6)
x#7$135A                            0.999999999999999   (obj:7)
x#7$127C                            0.999999999999998   (obj:7)
x#7$CPSN                            0.999999999999998   (obj:7)
x#8$135B                            0.999999999999999   (obj:8)
x#8$160                             0.999999999999997   (obj:8)
x#8$150A                            0.999999999999999   (obj:8)
```

X#1$21A ---- "1" quarter, and "21A" course

7. **Computer PROJECT 2:** *Automatically solving SUDOKU puzzles, predicting difficulty of Sudoku's:*

You probably know the famous sudoku puzzles, but just in case you do not, it consists of a A 9×9 matrix A is partitioned into nine 3×3 denoted $A_1, A_2, \ldots, A_9$. A few entries are filled in advanced, then a solution to the sudoku game is an assignment of integers from 1 to 9 to each (unassigned) entry of the matrix such that each row of A, each column of A and each A_1 contains every number from 1 to 9 exactly once.

- Formulate the problem of finding a solution for Sudoku as a discrete model. (HINT: Think of the assignment model, but use variables with 3 indices $x_{i,j,k}$ that takes value 1 or 0, depending on whether entry $A_{i,j}$ is assigned the number k.)
- Implement the model in ZIMPL/SCIP and use it to solve the following three SUDOKU puzzles, which one took longer?

			1	5	6			3
6	7	2		8		1		
				2	9	4	8	
9	3	4						1
			9	1			7	4
8			6		5			
		1		9	8		4	
5	4	9	2			3		
		6	4		1	2		9

9			8			2	7	3
			4					
	3	2						6
1		8						
		5		8		3		
	9				5		6	
				1		6	2	
					2	8		
5		3						9

8					7			
	3		1	5				7
							3	
9						1	6	8
		6			4			
1	7		5			4		
						2		1
2							7	
6			8				4	

```
param p := 3;

set L := { 1 .. p*p }; #create lineal elimination 1*9
set M := { 1 .. p}; #create square elimination 1*3


set F := { read "sudoku3.dat" as "<1n,2n,3n>"}; #read sudoku<"row","col","value">
var x[L * L * L] binary; #x[i,j,k] decision variable


#maximize cost: sum <i,j,k> in L*L*L : x[i,j,k]; objective

subto  bounds: sum <i,j,k> in L*L*L : x[i,j,k] == card(L*L); #not out-bounds
subto rows: forall <i,j> in L*L do sum <k> in L : x[i,j,k] == 1;
subto cols: forall <j,k> in L*L do sum <i> in L : x[i,j,k] == 1;
subto nums: forall <i,k> in L*L do sum <j> in L : x[i,j,k] == 1;

subto squares: forall <m,n,k> in M*M*L do
   sum <i,j> in M*M : x[(m-1)*p+i,(n-1)*p+j,k] == 1;

#Fix the fixed values
subto fixed: forall <i,j,k> in F do x[i,j,k] == 1;
```

Sudoku 1.-

```
read problem <sudoku.zpl>
============


base directory for ZIMPL parsing: </homes/home04/class/m160s1-6>

Multi:
|3|{<1,4,1>,<1,5,5>,<1,6,6>,<1,9,3>,<2,5,8>,<2,1,6>,<2,2,7>,<2,3,2>,<2,7,1>,<3,5,2>,<3,6,9>,<3,7,4>,<3,8,8
>,<4,9,1>,<4,1,9>,<4,2,3>,<4,3,4>,<5,4,9>,<5,5,1>,<5,9,4>,<5,8,7>,<6,4,6>,<6,6,5>,<6,1,8>,<7,5,9>,<7,6,8>,
<7,3,1>,<7,8,4>,<8,4,2>,<8,1,5>,<8,2,4>,<8,3,9>,<8,7,3>,<9,4,4>,<9,6,1>,<9,9,9>,<9,3,6>,<9,7,2>}
original problem has 729 variables (729 bin, 0 int, 0 impl, 0 cont) and 363 constraints
SCIP> optimize

presolving:
(round 1, fast)       729 del vars, 38 del conss, 0 add conss, 38 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd
conss, 0 impls, 0 clqs
presolving (2 rounds: 2 fast, 1 medium, 1 exhaustive):
 1420 deleted vars, 363 deleted constraints, 0 added constraints, 38 tightened bounds, 0 added holes, 0 changed
sides, 0 changed coefficients
 0 implications, 0 cliques
presolved problem has 0 variables (0 bin, 0 int, 0 impl, 0 cont) and 0 constraints
transformed objective value is always integral (scale: 1)
Presolving Time: 0.00

 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|  dualbound   |
primalbound  |  gap
t 0.0s|     1 |     0 |     0 |     - |1794k|   0 |   - |   0 |   0 |   0 |   0 |   0 |   0 |   0 |      --      | 0.000000e+00 |    Inf
```

```
  0.0s|     1 |     0 |     0 |     - |1793k|   0 |   - |   0 |   0 |   0 |   0 |   0 |   0 |   0 | 0.000000e+00 | 0.000000e+00 |
0.00%

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.01
Solving Nodes      : 1
Primal Bound       : +0.00000000000000e+00 (1 solutions)
Dual Bound         : +0.00000000000000e+00
Gap                : 0.00 %

SCIP> display solution

objective value:                                    0
x#1#1#4                                             1   (obj:0)
x#1#2#9                                             1   (obj:0)
x#1#3#8                                             1   (obj:0)
x#1#4#1                                             1   (obj:0)
x#1#5#5                                             1   (obj:0)
x#1#6#6                                             1   (obj:0)
x#1#7#7                                             1   (obj:0)
x#1#8#2                                             1   (obj:0)
x#1#9#3                                             1   (obj:0)
x#2#1#6                                             1   (obj:0)
x#2#2#7                                             1   (obj:0)
x#2#3#2                                             1   (obj:0)
x#2#4#3                                             1   (obj:0)
x#2#5#8                                             1   (obj:0)
x#2#6#4                                             1   (obj:0)
x#2#7#1                                             1   (obj:0)
x#2#8#9                                             1   (obj:0)
x#2#9#5                                             1   (obj:0)
x#3#1#1                                             1   (obj:0)
x#3#2#5                                             1   (obj:0)
x#3#3#3                                             1   (obj:0)
x#3#4#7                                             1   (obj:0)
x#3#5#2                                             1   (obj:0)
x#3#6#9                                             1   (obj:0)
x#3#7#4                                             1   (obj:0)
x#3#8#8                                             1   (obj:0)
x#3#9#6                                             1   (obj:0)
x#4#1#9                                             1   (obj:0)
x#4#2#3                                             1   (obj:0)
x#4#3#4                                             1   (obj:0)
x#4#4#8                                             1   (obj:0)
x#4#5#7                                             1   (obj:0)
x#4#6#2                                             1   (obj:0)
x#4#7#5                                             1   (obj:0)
x#4#8#6                                             1   (obj:0)
x#4#9#1                                             1   (obj:0)
x#5#1#2                                             1   (obj:0)
x#5#2#6                                             1   (obj:0)
x#5#3#5                                             1   (obj:0)
```

```
x#5#4#9                        1   (obj:0)
x#5#5#1                        1   (obj:0)
x#5#6#3                        1   (obj:0)
x#5#7#8                        1   (obj:0)
x#5#8#7                        1   (obj:0)
x#5#9#4                        1   (obj:0)
x#6#1#8                        1   (obj:0)
x#6#2#1                        1   (obj:0)
x#6#3#7                        1   (obj:0)
x#6#4#6                        1   (obj:0)
x#6#5#4                        1   (obj:0)
x#6#6#5                        1   (obj:0)
x#6#7#9                        1   (obj:0)
x#6#8#3                        1   (obj:0)
x#6#9#2                        1   (obj:0)
x#7#1#3                        1   (obj:0)
x#7#2#2                        1   (obj:0)
x#7#3#1                        1   (obj:0)
x#7#4#5                        1   (obj:0)
x#7#5#9                        1   (obj:0)
x#7#6#8                        1   (obj:0)
x#7#7#6                        1   (obj:0)
x#7#8#4                        1   (obj:0)
x#7#9#7                        1   (obj:0)
x#8#1#5                        1   (obj:0)
x#8#2#4                        1   (obj:0)
x#8#3#9                        1   (obj:0)
x#8#4#2                        1   (obj:0)
x#8#5#6                        1   (obj:0)
x#8#6#7                        1   (obj:0)
x#8#7#3                        1   (obj:0)
x#8#8#1                        1   (obj:0)
x#8#9#8                        1   (obj:0)
x#9#1#7                        1   (obj:0)
x#9#2#8                        1   (obj:0)
x#9#3#6                        1   (obj:0)
x#9#4#4                        1   (obj:0)
x#9#5#3                        1   (obj:0)
x#9#6#1                        1   (obj:0)
x#9#7#2                        1   (obj:0)
x#9#8#5                        1   (obj:0)
x#9#9#9                        1   (obj:0)
```

SUDOKU 2.-

```
SCIP> read sudoku.zpl

read problem <sudoku.zpl>
============
```

```
base directory for ZIMPL parsing: </homes/home04/class/m160s1-6>

original problem has 729 variables (729 bin, 0 int, 0 impl, 0 cont) and 350 constraints
SCIP> optimize

presolving:
(round 1, fast)       605 del vars, 25 del conss, 0 add conss, 25 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd
conss, 0 impls, 152 clqs
(round 2, fast)       1309 del vars, 247 del conss, 0 add conss, 32 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd
conss, 0 impls, 0 clqs
presolving (3 rounds: 3 fast, 1 medium, 1 exhaustive):
 1386 deleted vars, 350 deleted constraints, 0 added constraints, 32 tightened bounds, 0 added holes, 0 changed
sides, 0 changed coefficients
 0 implications, 0 cliques
presolved problem has 0 variables (0 bin, 0 int, 0 impl, 0 cont) and 0 constraints
transformed objective value is always integral (scale: 1)
Presolving Time: 0.01

 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|  dualbound   |
primalbound  |  gap
t 0.0s|     1 |     0 |     0 |     - |1788k|   0 |   - |   0 |   0 |   0 |   0 |   0 |   0 |   0 |      --      | 0.000000e+00 |    Inf
  0.0s|     1 |     0 |     0 |     - |1787k|   0 |   - |   0 |   0 |   0 |   0 |   0 |   0 |   0 | 0.000000e+00 | 0.000000e+00 |
0.00%

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.02
Solving Nodes      : 1
Primal Bound       : +0.00000000000000e+00 (1 solutions)
Dual Bound         : +0.00000000000000e+00
Gap                : 0.00 %

SCIP> display solution

objective value:                                    0
x#1#1#9                                             1   (obj:0)
x#1#2#4                                             1   (obj:0)
x#1#3#1                                             1   (obj:0)
x#1#4#8                                             1   (obj:0)
x#1#5#5                                             1   (obj:0)
x#1#6#6                                             1   (obj:0)
x#1#7#2                                             1   (obj:0)
x#1#8#7                                             1   (obj:0)
x#1#9#3                                             1   (obj:0)
x#2#1#7                                             1   (obj:0)
x#2#2#5                                             1   (obj:0)
x#2#3#6                                             1   (obj:0)
x#2#4#4                                             1   (obj:0)
x#2#5#2                                             1   (obj:0)
x#2#6#3                                             1   (obj:0)
x#2#7#9                                             1   (obj:0)
x#2#8#8                                             1   (obj:0)
```

x#2#9#1	1	(obj:0)
x#3#1#8	1	(obj:0)
x#3#2#3	1	(obj:0)
x#3#3#2	1	(obj:0)
x#3#4#1	1	(obj:0)
x#3#5#9	1	(obj:0)
x#3#6#7	1	(obj:0)
x#3#7#5	1	(obj:0)
x#3#8#4	1	(obj:0)
x#3#9#6	1	(obj:0)
x#4#1#1	1	(obj:0)
x#4#2#6	1	(obj:0)
x#4#3#8	1	(obj:0)
x#4#4#9	1	(obj:0)
x#4#5#3	1	(obj:0)
x#4#6#4	1	(obj:0)
x#4#7#7	1	(obj:0)
x#4#8#5	1	(obj:0)
x#4#9#2	1	(obj:0)
x#5#1#2	1	(obj:0)
x#5#2#7	1	(obj:0)
x#5#3#5	1	(obj:0)
x#5#4#6	1	(obj:0)
x#5#5#8	1	(obj:0)
x#5#6#1	1	(obj:0)
x#5#7#3	1	(obj:0)
x#5#8#9	1	(obj:0)
x#5#9#4	1	(obj:0)
x#6#1#3	1	(obj:0)
x#6#2#9	1	(obj:0)
x#6#3#4	1	(obj:0)
x#6#4#2	1	(obj:0)
x#6#5#7	1	(obj:0)
x#6#6#5	1	(obj:0)
x#6#7#1	1	(obj:0)
x#6#8#6	1	(obj:0)
x#6#9#8	1	(obj:0)
x#7#1#4	1	(obj:0)
x#7#2#8	1	(obj:0)
x#7#3#7	1	(obj:0)
x#7#4#3	1	(obj:0)
x#7#5#1	1	(obj:0)
x#7#6#9	1	(obj:0)
x#7#7#6	1	(obj:0)
x#7#8#2	1	(obj:0)
x#7#9#5	1	(obj:0)
x#8#1#6	1	(obj:0)
x#8#2#1	1	(obj:0)
x#8#3#9	1	(obj:0)
x#8#4#5	1	(obj:0)
x#8#5#4	1	(obj:0)
x#8#6#2	1	(obj:0)

x#8#7#8 1 (obj:0)
x#8#8#3 1 (obj:0)
x#8#9#7 1 (obj:0)
x#9#1#5 1 (obj:0)
x#9#2#2 1 (obj:0)
x#9#3#3 1 (obj:0)
x#9#4#7 1 (obj:0)
x#9#5#6 1 (obj:0)
x#9#6#8 1 (obj:0)
x#9#7#4 1 (obj:0)
x#9#8#1 1 (obj:0)
x#9#9#9 1 (obj:0)

SUDOKU 3.-

SCIP> read sudoku.zpl

read problem <sudoku.zpl>
============

base directory for ZIMPL parsing: </homes/home04/class/m160s1-6>

original problem has 729 variables (729 bin, 0 int, 0 impl, 0 cont) and 349 constraints
SCIP> optimize

presolving:
(round 1, fast) 530 del vars, 24 del conss, 0 add conss, 24 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd conss, 0 impls, 195 clqs
(round 2, fast) 1143 del vars, 188 del conss, 0 add conss, 27 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd conss, 0 impls, 101 clqs
(round 3, fast) 1233 del vars, 265 del conss, 0 add conss, 27 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd conss, 0 impls, 74 clqs
(round 4, exhaustive) 1233 del vars, 274 del conss, 0 add conss, 27 chg bounds, 0 chg sides, 0 chg coeffs, 0 upgd conss, 0 impls, 74 clqs
(round 5, exhaustive) 1233 del vars, 274 del conss, 0 add conss, 27 chg bounds, 0 chg sides, 0 chg coeffs, 74 upgd conss, 0 impls, 74 clqs
(round 6, exhaustive) 1306 del vars, 274 del conss, 0 add conss, 27 chg bounds, 0 chg sides, 0 chg coeffs, 74 upgd conss, 0 impls, 0 clqs
presolving (7 rounds: 7 fast, 4 medium, 4 exhaustive):
1354 deleted vars, 349 deleted constraints, 0 added constraints, 27 tightened bounds, 0 added holes, 0 changed sides, 0 changed coefficients
0 implications, 0 cliques
presolved problem has 0 variables (0 bin, 0 int, 0 impl, 0 cont) and 0 constraints
transformed objective value is always integral (scale: 1)
Presolving Time: 0.02

```
 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|  dualbound   |
primalbound  |  gap
t 0.0s|     1 |     0 |     0 |     - |1856k|   0 |   - |   0 |   0 |   0 |   0 |   0 |   0 |   0 |      --      | 0.000000e+00 |    Inf
  0.0s|     1 |     0 |     0 |     - |1855k|   0 |   - |   0 |   0 |   0 |   0 |   0 |   0 |   0 | 0.000000e+00 | 0.000000e+00 |
0.00%
```

```
SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.03
Solving Nodes      : 1
Primal Bound       : +0.00000000000000e+00 (1 solutions)
Dual Bound         : +0.00000000000000e+00
Gap                : 0.00 %

SCIP> display solution

objective value:                                    0
x#1#1#8                                             1   (obj:0)
x#1#2#2                                             1   (obj:0)
x#1#3#5                                             1   (obj:0)
x#1#4#6                                             1   (obj:0)
x#1#5#3                                             1   (obj:0)
x#1#6#7                                             1   (obj:0)
x#1#7#9                                             1   (obj:0)
x#1#8#1                                             1   (obj:0)
x#1#9#4                                             1   (obj:0)
x#2#1#4                                             1   (obj:0)
x#2#2#3                                             1   (obj:0)
x#2#3#9                                             1   (obj:0)
x#2#4#1                                             1   (obj:0)
x#2#5#5                                             1   (obj:0)
x#2#6#8                                             1   (obj:0)
x#2#7#6                                             1   (obj:0)
x#2#8#2                                             1   (obj:0)
x#2#9#7                                             1   (obj:0)
x#3#1#7                                             1   (obj:0)
x#3#2#6                                             1   (obj:0)
x#3#3#1                                             1   (obj:0)
x#3#4#4                                             1   (obj:0)
x#3#5#9                                             1   (obj:0)
x#3#6#2                                             1   (obj:0)
x#3#7#8                                             1   (obj:0)
x#3#8#3                                             1   (obj:0)
x#3#9#5                                             1   (obj:0)
x#4#1#9                                             1   (obj:0)
x#4#2#5                                             1   (obj:0)
x#4#3#4                                             1   (obj:0)
x#4#4#2                                             1   (obj:0)
x#4#5#7                                             1   (obj:0)
x#4#6#3                                             1   (obj:0)
x#4#7#1                                             1   (obj:0)
x#4#8#6                                             1   (obj:0)
x#4#9#8                                             1   (obj:0)
x#5#1#3                                             1   (obj:0)
x#5#2#8                                             1   (obj:0)
x#5#3#6                                             1   (obj:0)
x#5#4#9                                             1   (obj:0)
x#5#5#1                                             1   (obj:0)
```

x#5#6#4	1	(obj:0)
x#5#7#7	1	(obj:0)
x#5#8#5	1	(obj:0)
x#5#9#2	1	(obj:0)
x#6#1#1	1	(obj:0)
x#6#2#7	1	(obj:0)
x#6#3#2	1	(obj:0)
x#6#4#5	1	(obj:0)
x#6#5#8	1	(obj:0)
x#6#6#6	1	(obj:0)
x#6#7#4	1	(obj:0)
x#6#8#9	1	(obj:0)
x#6#9#3	1	(obj:0)
x#7#1#5	1	(obj:0)
x#7#2#4	1	(obj:0)
x#7#3#3	1	(obj:0)
x#7#4#7	1	(obj:0)
x#7#5#6	1	(obj:0)
x#7#6#9	1	(obj:0)
x#7#7#2	1	(obj:0)
x#7#8#8	1	(obj:0)
x#7#9#1	1	(obj:0)
x#8#1#2	1	(obj:0)
x#8#2#9	1	(obj:0)
x#8#3#8	1	(obj:0)
x#8#4#3	1	(obj:0)
x#8#5#4	1	(obj:0)
x#8#6#1	1	(obj:0)
x#8#7#5	1	(obj:0)
x#8#8#7	1	(obj:0)
x#8#9#6	1	(obj:0)
x#9#1#6	1	(obj:0)
x#9#2#1	1	(obj:0)
x#9#3#7	1	(obj:0)
x#9#4#8	1	(obj:0)
x#9#5#2	1	(obj:0)
x#9#6#5	1	(obj:0)
x#9#7#3	1	(obj:0)
x#9#8#4	1	(obj:0)
x#9#9#9	1	(obj:0)