Federal State Autonomous Educational Institution of Higher Education "Moscow Institute of Physics and Technology (National Research University)"

APPROVED
Head of the Phystech School of Applied Mathematics and Informatics
A.M. Raygorodskiy

Work program of the course (training module)

course:	Constraint Programming/Программирование в ограничениях
major:	Applied Mathematics and Informatics
specialization:	Computer Science/Информатика
	Phystech School of Applied Mathematics and Informatics
	Chair of Discrete Mathematics
term:	2
qualification:	Bachelor

Semester, form of interim assessment: 4 (spring) - Grading test

Academic hours: 60 AH in total, including:
- lectures: 0 AH.
- seminars: 60 AH.
- laboratory practical: 0 AH.

Independent work: 75 AH.

In total: 135 AH, credits in total: 3

Author of the program: A.B. Daynyak, candidate of physics and mathematical sciences, associate professor, associate professor

The program was discussed at the Chair of Discrete Mathematics 05.03.2020

Annotation

The course is devoted to one of the modern programming paradigms that has common flavour to functional and logical programming, and often eludes the standard computer science curriculum, yet is highly practical and beneficial for the applied mathematical culture of the students: the Constraint Programming (CP). In constraint programming, unlike the ubiquitous imperative programming, instead of describing the sequence of elementary operations needed to get a desired result (object/configuration), we rather describe the set of elementary constraints — the properties that the object should have to be called the result. Constructing of the configuration that meets all the requirements is then left to a special program called solver (analogous to interpreter in imperative programming). The programmer who is working in CP paradigm does not usually have direct control over the time complexity of the computation, but there are many standard proven techniques that improve the efficiency of the programs (models as they are called in CP) by orders of magnitude. The course is aimed at familiarizing the students with these tools, and it often implies some non-trivial mathematical reasoning. For practicing working in CP paradigm we use one of the standard modern languages MiniZinc.

1. Study objective

Purpose of the course

The course is devoted to Constraint Programming (CP) - a discipline lying at the junction of mathematical modeling and computer programming, which can be considered a separate programming paradigm, close to logical and functional programming and clearly different from the most common imperative programming paradigm. Instead of describing elementary operations leading to the achievement of the result, that is, some object / configuration.

The main task in CP is to describe what elementary conditions an object must satisfy in order to be considered a result.

Tasks of the course

The course is designed to provide an opportunity to practice modeling simplified and real discrete optimization problems in one of the standard modern CP-languages MiniZinc.

2. List of the planned results of the course (training module), correlated with the planned results of the mastering the educational program

Mastering the discipline is aimed at the formation of the following competencies:

Code and the name of the competence	Competency indicators
Gen.Pro.C-2 Use modern IT and software tools to perform professional tasks in compliance with information security requirements	Gen.Pro.C-2.1 Apply modern computing tools and Internet services in professional settings
	Gen.Pro.C-2.2 Apply numerical mathematical methods and use software applications for scientific problem-solving in professional settings
	Gen.Pro.C-2.3 Fulfill basic information security requirements
Pro.C-1 Assign, formalize, and solve tasks, develop and research mathematical models of studied phenomena and processes, systematically analyze scientific problems, obtain new scientific outcomes	Pro.C-1.1 Locate, analyze, and summarize information on current research findings within the subject area
	Pro.C-1.2 Make hypotheses, build mathematical models of the studied phenomena and processes, evaluate the quality of the developed model
	ПК-1.3 Apply theoretical and/or experimental research methods to a specific scientific task and interpret the obtained results
Pro.C-2 Conduct scientific research and testing independently or as a member (leader) of a small research team	Pro.C-2.2 Conduct scientific research independently or as a member (leader) of a small research team
	Pro.C-2.1 Apply the principles of scientific work, methods of collecting and analyzing obtained data and ways of argumentation
	Pro.C-2.3 Present research results through scientific publications and participation in conferences

3. List of the planned results of the course (training module)

As a result of studying the course the student should:

know:

- in CP, the main task is to describe what elementary conditions an object must satisfy in order to be considered a result.

be able to:

- simulation of simplified and real discrete optimization problems in one of the standard modern CP-languages MiniZinc.

master:

- basic syntax of the MiniZinc language. Definition of variables and constants. Arrays. Model + data. Limitations. Output format.

4. Content of the course (training module), structured by topics (sections), indicating the number of allocated academic hours and types of training sessions

4.1. The sections of the course (training module) and the complexity of the types of training sessions

№	Topic (section) of the course	Types of training sessions, including independent work			
		Lectures	Seminars	Laboratory practical	Independent work
1	Constraint programming: its differences with imperative programming.		8		9
2	Terminology and mathematical formalization of constraint satisfaction and optimization.		8		9
3	MiniZinc and FlatZinc syntax basics.		7		8
4	Types of solvers: CP solvers, MILP solvers.		6		8
5	Global constraints		5		8
6	Linear Programming as a modeling tool and mathematical subject		8		9
7	Symmetries of the search space. Breaking symmetries.		8		8
8	Search mechanics of CP solvers.		5		8
9	Industrial tools for CP and optimization.		5		8
AH in total			60		75
Exam preparation		0 AH.			
Total complexity		135 AH., credits in total 3			

4.2. Content of the course (training module), structured by topics (sections)

Semester: 4 (Spring)

1. Constraint programming: its differences with imperative programming.

Models vs. programs, solvers vs. interpreters, constraints vs. commands/instructions, quantifiers vs. loops, etc.

2. Terminology and mathematical formalization of constraint satisfaction and optimization.

Feasible solutions, optimal solutions. Search space. Reduction of optimization to constraint satisfaction.

3. MiniZinc and FlatZinc syntax basics.

Variable and constant definitions. Models and data files. Constraints. Output statements.

4. Types of solvers: CP solvers, MILP solvers.

Their strengths and limitations. Examples of the same problem modelled differently for CP vs MILP solvers based on N-Queens problem.

5. Global constraints

Alldifferent, increasing and other typical global constraints. Implementation of alldifferent constraint.

6. Linear Programming as a modeling tool and mathematical subject

Modeling logical constraints with linear constraints and integer variables. Duality in linear programming. Duality as certifiability.

7. Symmetries of the search space. Breaking symmetries.

Symmetry breaking constraints vs. redundant constraints.

8. Search mechanics of CP solvers.

Variable choice and value choice. Restarts. Search annotations.

9. Industrial tools for CP and optimization.

Google OR tools. Interfacing with Python. Practice of constraint programming.

5. Description of the material and technical facilities that are necessary for the implementation of the educational process of the course (training module)

The classroom is equipped with personal computers, a multimedia projector and a screen.

6. List of the main and additional literature, that is necessary for the course (training module) mastering

Main literature

1. MATLAB 7 [Текст] : программирование, численные методы / Ю. Л. Кетков, А. Ю. Кетков, М. М. Шульц .— СПб. : БХВ-Петербург, 2005 .— 737 с.
2. Линейное программирование [Текст] / Ф. П. Васильев, А. Ю. Иваницкий - М.Факториал Пресс,2008

Additional literature

1. MATLAB 7 [Текст] : в подлиннике : наиболее полное руководство / И. Е. Ануфриев, А. Б. Смирнов, Е. Н. Смирнова .— СПб. : БХВ-Петербург, 2005 .— 1104 с.

7. List of web resources that are necessary for the course (training module) mastering

http://www.mou.mipt.ru
http://www.exponenta.ru/educat/free/matlab/gs.pdf

8. List of information technologies used for implementation of the educational process, including a list of software and information reference systems (if necessary)

The classes use multimedia technologies, including demonstration of presentations, as well as the MiniZinc CP language software development tool.

9. Guidelines for students to master the course

1. It is recommended to successfully pass test papers, as this simplifies the final certification in the subject.
2. To prepare for the final certification in the subject, it is best to use the lecture materials.

SUPPLEMENT

Assessment funds for course (training module)

major:	Applied Mathematics and Informatics
specialization:	Computer Science/Информатика
	Phystech School of Applied Mathematics and Informatics
	Chair of Discrete Mathematics
term:	2
qualification:	Bachelor

Semester, form of interim assessment: 4 (spring) - Grading test

Author:	A.B. Daynyak, candidate of physics and mathematical sciences, associate professor, associate professor

1. Competencies formed during the process of studying the course

Code and the name of the competence	Competency indicators
Gen.Pro.C-2 Use modern IT and software tools to perform professional tasks in compliance with information security requirements	Gen.Pro.C-2.1 Apply modern computing tools and Internet services in professional settings
	Gen.Pro.C-2.2 Apply numerical mathematical methods and use software applications for scientific problem-solving in professional settings
	Gen.Pro.C-2.3 Fulfill basic information security requirements
Pro.C-1 Assign, formalize, and solve tasks, develop and research mathematical models of studied phenomena and processes, systematically analyze scientific problems, obtain new scientific outcomes	Pro.C-1.1 Locate, analyze, and summarize information on current research findings within the subject area
	Pro.C-1.2 Make hypotheses, build mathematical models of the studied phenomena and processes, evaluate the quality of the developed model
	ПК-1.3 Apply theoretical and/or experimental research methods to a specific scientific task and interpret the obtained results
Pro.C-2 Conduct scientific research and testing independently or as a member (leader) of a small research team	Pro.C-2.2 Conduct scientific research independently or as a member (leader) of a small research team
	Pro.C-2.1 Apply the principles of scientific work, methods of collecting and analyzing obtained data and ways of argumentation
	Pro.C-2.3 Present research results through scientific publications and participation in conferences

2. Competency assessment indicators

As a result of studying the course the student should:

know:

- in CP, the main task is to describe what elementary conditions an object must satisfy in order to be considered a result.

be able to:

- simulation of simplified and real discrete optimization problems in one of the standard modern CP-languages MiniZinc.

master:

- basic syntax of the MiniZinc language. Definition of variables and constants. Arrays. Model + data. Limitations. Output format.

3. List of typical control tasks used to evaluate knowledge and skills

The current control consists of two tests per semester, as well as oral delivery of assignments for independent decision. Evaluation criteria are attached. Also attached is an example of a test task and several problems for independent solution on various topics at the end of the program.

4. Evaluation criteria

1. What feature of imperative programming is roughly analogous to the for loop in imperative programming?
2. Why are exists and forall quantifiers different in complexity in ILP modeling?
3. What is unrolling when converting MiniZinc program into FlatZinc?
4. Provide an example of a redundant constraint and a symmetry breaking constraint.
5. When do we say that one constraint covers the other one?
6. Formulate a dual LP for the given primal LP.

7. Provide an example use case of an alldifferent constraint. What would be the equivalent forall constraint?
8. Provide the example of interchangeably using if-then-else and Boolean implication.
9. What syntactic structures on MiniZinc can we use to influence the way that it searches for a solution to the model? Provide a use case example.
10. Which of the solvers COIN-BC, Gecode, Chuffed would better solve an ILP model and why.
11. How one can experimentally prove that for a given model the given constraint is redundant?

Assessment “excellent (10)” is given to a student who has displayed comprehensive, systematic and deep knowledge of the educational program material, has independently performed all the tasks stipulated by the program, has deeply studied the basic and additional literature recommended by the program, has been actively working in the classroom, and understands the basic scientific concepts on studied discipline, who showed creativity and scientific approach in understanding and presenting educational program material, whose answer is characterized by using rich and adequate terms, and by the consistent and logical presentation of the material;

Assessment “excellent (9)” is given to a student who has displayed comprehensive, systematic knowledge of the educational program material, has independently performed all the tasks provided by the program, has deeply mastered the basic literature and is familiar with the additional literature recommended by the program, has been actively working in the classroom, has shown the systematic nature of knowledge on discipline sufficient for further study, as well as the ability to amplify it on one’s own, whose answer is distinguished by the accuracy of the terms used, and the presentation of the material in it is consistent and logical;

Assessment “excellent (8)” is given to a student who has displayed complete knowledge of the educational program material, does not allow significant inaccuracies in his answer, has independently performed all the tasks stipulated by the program, studied the basic literature recommended by the program, worked actively in the classroom, showed systematic character of his knowledge of the discipline, which is sufficient for further study, as well as the ability to amplify it on his own;

Assessment “good (7)” is given to a student who has displayed a sufficiently complete knowledge of the educational program material, does not allow significant inaccuracies in the answer, has independently performed all the tasks provided by the program, studied the basic literature recommended by the program, worked actively in the classroom, showed systematic character of his knowledge of the discipline, which is sufficient for further study, as well as the ability to amplify it on his own;

Assessment “good (6)” is given to a student who has displayed a sufficiently complete knowledge of the educational program material, does not allow significant inaccuracies in his answer, has independently carried out the main tasks stipulated by the program, studied the basic literature recommended by the program, showed systematic character of his knowledge of the discipline, which is sufficient for further study;

Assessment “good (5)” is given to a student who has displayed knowledge of the basic educational program material in the amount necessary for further study and future work in the profession, who while not being sufficiently active in the classroom, has nevertheless independently carried out the main tasks stipulated by the program, mastered the basic literature recommended by the program, made some errors in their implementation and in his answer during the test, but has the necessary knowledge for correcting these errors by himself;

Assessment “satisfactory (4)” is given to a student who has discovered knowledge of the basic educational program material in the amount necessary for further study and future work in the profession, who while not being sufficiently active in the classroom, has nevertheless independently carried out the main tasks stipulated by the program, learned the main literature but allowed some errors in their implementation and in his answer during the test, but has the necessary knowledge for correcting these errors under the guidance of a teacher;

Assessment “satisfactory (3)” is given to a student who has displayed knowledge of the basic educational program material in the amount necessary for further study and future work in the profession, not showed activity in the classroom, independently fulfilled the main tasks envisaged by the program, but allowed errors in their implementation and in the answer during the test, but possessing necessary knowledge for elimination under the guidance of the teacher of the most essential errors;

Assessment “unsatisfactory (2)” is given to a student who showed gaps in knowledge or lack of knowledge on a significant part of the basic educational program material, who has not performed independently the main tasks demanded by the program, made fundamental errors in the fulfillment of the tasks stipulated by the program, who is not able to continue his studies or start professional activities without additional training in the discipline in question;

Assessment “unsatisfactory (1)” is given to a student when there is no answer (refusal to answer), or when the submitted answer does not correspond at all to the essence of the questions contained in the task.

5. Methodological materials defining the procedures for the assessment of knowledge, skills, abilities and/or experience

The test is carried out on the basis of the current performance and delivery of tasks and term paper. Submission of term paper is carried out in the form of an oral report for 15-20 minutes. The topic of the course work is selected by the student, but must be previously agreed with the teacher and must comply with the course program. In the course work, the solution to the applied problem using mathematical modeling methods implemented in the Matlab environment should be presented (other software packages can be used by agreement with the teacher).

ENG
\begin{problem}
What is the result of running the following model in MiniZinc? How would you explain it?
\begin{lstlisting}

```
var int: x = 4;
var int: y = 5;
var int: t;

constraint t = x;
constraint x = y;
constraint y = t;

output [show(x), show(y)];
```

\end{lstlisting}
\end{problem}

\begin{problem}
Important: in the current problem use only Gecode solver! In class we have looked at the following model for computing Fibonacci numbers:
\begin{lstlisting}

```
int: N = 30;
array [0..N] of var int: fib;

constraint
  forall(i in 2..N)(
    fib[i] = fib[i-1] + fib[i-2]
  );

constraint
  fib[0] = 1;
constraint
  fib[N] = 1346269;

output [
  "The result is: ",
  join(
    " ",
    [show(x) | x in fib]
  )
];
```

\end{lstlisting}

Experimentally find the largest such \verb"N" for which specifying just the first and the last values of the Fibonacci array allows us to compute all the other elements of the array within one second. Then provide an additional knowledge to the solver, that all the Fibonacci numbers are positive:
\begin{lstlisting}

```
constraint forall(i in 0..N)(
  fib[i] >= 0
);
```

\end{lstlisting}
How does the maximal \verb"N" change, what value it not takes?
\end{problem}

\begin{problem}

Consider the following linear constraints:

```
\begin{eqnarray*}
x_1 - 2x_2 + 6 &\le& 0,\\
-3x_1 + 5x_2 &\ge& -15,\\
-3x_1 + 6x_2 &\le& 18,\\
-x_1 + x_2 &\le& 3,\\
4x_1 - 3x_2 &\le& 7,\\
x_1,x_2 &\ge& 0.
\end{eqnarray*}
```

Find [with different models] minimum and maximum values for \(f(x_1, x_2) = 2x_1 + x_2\) under these constraints, and make your model output show which values of \(x_1, x_2\) result in min/max values. Try solving the problem with different solvers.
\end{problem}

\begin{problem}\label{pro-1}
Consider the following model for NQUEENS problem, in which we use “coordinates” of the queens as our decision variables:

```
\begin{lstlisting}
int: N = 8;
set of int: QUEEN = 1..N;
set of int: ROW = 1..N;
set of int: COLUMN = 1..N;

array [QUEEN] of var ROW: row;
array [QUEEN] of var COLUMN: column;

constraint forall(q1 in QUEEN, q2 in QUEEN where q2 != q1)(
  row[q1] != row[q2]
  /\
  column[q1] != column[q2]
  /\
  row[q1] - column[q1] != row[q2] - column[q2]
  /\
  row[q1] + column[q1] != row[q2] + column[q2]
);
\end{lstlisting}
```

Develop a proper \verb"output" statement for this model so that we can get the visual representation of the board as we have in other models developed in class. You can use the following template as a reference, filling in all \verb"???" placeholders:

```
\begin{lstlisting}
output [
  join(
  "\n", [
    join(
      " ", [
        if exists(???)(fix(???) = i /\ fix(???) = j)
        then "X"
        else "O"
        endif
      | j in COLUMN]
    )
  | i in ROW]
  )
];
\end{lstlisting}
```

Why do we have to use \verb"fix" here?

Also recall that in case of satisfaction problems we can add an optional \verb"solve satisfy;" statement. Add it to the current model.

As an answer to this problem submit a complete model including your \verb"output" statement.
\end{problem}

\newpage
\begin{problem}
Note how we use named ranges \verb"QUEEN", \verb"ROW" and \verb"COLUMN" to improve the readability of the model in Problem~\ref{pro-1}. Rework the first “board-centric model with Boolean decision variables” that we have designed in class, by introducing constant named ranges to it:
\begin{lstlisting}

```
int: N = 22;

array[1..N, 1..N] of var bool: board;

constraint
  forall(i in 1..N)(
    exists(j in 1..N)(
      board[i,j] = true
    )
  );

constraint
  forall(i in 1..N, j in 1..N, iother in 1..N where i != iother)(
    if board[i,j] = true then board[iother,j] = false endif
  );

constraint
  forall(i in 1..N, j in 1..N, jother in 1..N where j != jother)(
    board[i,j] = true -> board[i,jother] = false
  );


constraint
  forall(
    i in 1..N, j in 1..N, c in (-min(i, j)+1)..min(N-i, N-j)
    where c != 0
  )(
    board[i,j] = true -> board[i+c, j+c] = false
  );

constraint
  forall(
    i in 1..N, j in 1..N, c in max(1-i, j-N)..min(j-1, N-i)
    where c != 0
  )(
    board[i,j] = true -> board[i+c, j-c] = false
  );


solve satisfy;
```

```
output [
  join(
    "\n",
    [join(" ", [
      if fix(board[i,j]) = true then "X" else "O" endif
    | j in 1..N]) | i in 1..N]
  )
];
\end{lstlisting}

As an answer to this problem submit your reworked model.
\end{problem}


\newpage
\begin{problem}
Consider the following model that we worked on in class that uses 0/1-variables instead of pure Boolean variables:
\begin{lstlisting}
int: N = 8;

array[1..N, 1..N] of var 0..1: board;

% Horizontal line constraints:
constraint
  forall(i in 1..N)(
    sum(j in 1..N)(board[i,j]) = 1
  );

% Vertical line constraints:
constraint
  forall(j in 1..N)(
    sum(i in 1..N)(board[i,j]) = 1
  );

% Primary diagonals line constraints:
constraint
  forall(c in 1-N..N-1)(
    sum(i in max(1, 1+c)..min(N, N+c))(board[i, i-c]) <= 1
  );

% Secondary diagonals line constraints:
constraint
  forall(
    i in 1..N, j in 1..N, c in max(1-i, j-N)..min(j-1, N-i)
    where c != 0
  )(
    board[i,j] = 1 -> board[i+c, j-c] = 0
  );

solve satisfy;

output [
  join(
    "\n",
    [join(" ", [
      if fix(board[i,j]) = true then "X" else "O" endif
    | j in 1..N]) | i in 1..N]
```

```
      )
    ];
    \end{lstlisting}

    \begin{subproblems}
    \item Consider \verb"forall(c in 1-N..N-1)" in “Primary diagonals line constraints”.
Take \(N=4\) and sketch what diagonal of the board is covered by every fixed value of
\verb"c". Can the range of \verb"c" be reduced to smaller one? Explain your answer.
    \item In class we have not finished the transition from “logic constraints” (involving
\verb"->") to “arithmetic constraints” (like \verb"sum(…) <= 1") in “Secondary diagonals
line constraints”. Complete this transition and try COIN-BC on the resulting model which will
now have only linear constraints. Can you now solve the NQUEENS problem for \(N=50\) in
under 1 second?
    \end{subproblems}
    \end{problem}



    \begin{problem}
    Consider the following dummy model whose only purpose is to output some funny
things into console:
    \begin{lstlisting}
    % Assume that n is an even number from 2 to 20
    int: n = 8;
    solve satisfy;
    output […];
    \end{lstlisting}
    Implement a MiniZinc \verb©output […]© statement that will produce the following
output to MiniZinc console:
    \begin{lstlisting}
    +
    ++
    +++
    ++++
        ++++
         +++
          ++
           +
    \end{lstlisting}
    So the output has \verb©n© lines and uses only space and plus symbols, producing
the figure above. Here is also an example for \verb©n = 10©:
    \begin{lstlisting}
    +
    ++
    +++
    ++++
    +++++
         +++++
          ++++
           +++
            ++
             +
    \end{lstlisting}
    \end{problem}

    \begin{problem}
    The KNAPSACK problem is stated as follows. There are several items with known
varying weights and values. There is a knapsack of known maximal weight capacity. The
goal is to pack some of the items into the knapsack without breaking it (due to overweight)
so that the total value of packed items is maximized. Your goal is to solve the problem with
```

MiniZinc. Let \verb"value" be the array that stores the values of the items and \verb"weight" be the array storing the weights, and let \verb"MAX_WEIGHT" be the maximal weight capacity of the knapsack:

```
\begin{lstlisting}
int: N = 15;
set of int: ITEM = 1..N;

int: MAX_WEIGHT = 150;

array[ITEM] of int:
   value = [1, 1, 2, 5, 3, 4, 5, 2, 1, 1, 2, 2, 3, 4, 4];
array[ITEM] of int:
   weight = [5, 8, 9, 30, 20, 15, 35, 13, 17, 12, 9, 6, 5, 27, 5];
\end{lstlisting}
```

\begin{subproblems}

\item Implement a model that has a "pure Boolean" (i.e. \verb"bool" in MiniZinc) decision variable for each item (it will be convenient to have an array of these variables), each of them being an indicator of taking the corresponding item into the knapsack. Use \verb"sum(i in ITEM where …)(…)" to compute the weight of the items the decision variables of which are set to \verb"true".

\item Implement an LP model for the KNAPSACK problem as follows. Introduce 0/1 decision variable for each item (it will be convenient to have an array of these variables), each of them being an indicator of taking the corresponding item into the knapsack. Express the objective function in this optimization problem as a linear combination of the decision variables. Also express the "non-overweighting" constraint as a linear inequality involving the decision variable. Thus you will be able to run COIN-BC solver on your model.

\end{subproblems}

\end{problem}

\begin{problem}\label{cryptarithmetic}

Solve the following cryptarithmetic puzzle\footnote{See the definition of cryptarithmetic puzzles \link{https://en.wikipedia.org/wiki/Verbal_arithmetic}{here}.} by modeling it with MiniZinc (introduce the variables according to unknown letters in the puzzle, and encode the constraint that matches the puzzle): \verb©MIPT + START = PARTY©

\end{problem}

\begin{problem}

Suppose variables \verb"x" and \verb"y" are defined as follows:

```
\begin{lstlisting}
var float: x;
var float: y;
\end{lstlisting}
```

Out of the following list of constraints one of them is covered by the other two. Which one is it? Explain the answer.

```
\begin{lstlisting}
constraint 5*x + 4*y >= 20;
constraint 10*x + y >= 10;
constraint 8*x + 9*y >= 70;
\end{lstlisting}
```

\end{problem} % Answer: 5*x + 4*y >= 20;

\begin{problem}

Consider the following model of N-queens that we implemented in class (with named ranges introduced for readability):

```
\begin{lstlisting}
int: N = 100;
set of int: ROW = 1..N;
set of int: COLUMN = 1..N;

array[ROW, COLUMN] of var 0..1: board;

constraint forall(r in ROW)(sum(c in COLUMN)(board[r,c]) = 1);
constraint forall(c in COLUMN)(sum(r in ROW)(board[r,c]) = 1);

constraint forall(c in 1-N..N-1)(
   sum(i in max(1, 1+c)..min(N, N+c))(board[i, i-c]) <= 1
  );

constraint forall(c in 1-N..N-1)(
   sum(i in max(1, 1+c)..min(N, N+c))(board[i, N+1-i+c]) <= 1
  );

output [join("\n", [join(" ", [
     if fix(board[r, c]) = true then "X" else "-" endif
   | c in COLUMN]) | r in ROW])];
\end{lstlisting}
```

Which of the following constraints is/are redundant (i.e. does not change the set of all feasible solutions to the problem\footnote{Actually, we can also define a constraint as being redundant if it is covered by the constraints that are already present in your model}):
\begin{subproblems}

```
\item \begin{lstlisting}
constraint sum(r in ROW, c in COLUMN)(board[r,c]) <= N;
\end{lstlisting}

\item \begin{lstlisting}
constraint exists(r in ROW)(sum(c in COLUMN)(board[r,c]) = 1);
\end{lstlisting}

\item \begin{lstlisting}
constraint sum(i in 1..N)(board[i,i]) >= 1;
\end{lstlisting}
```

\end{subproblems}

Explain your answer. I.e. if the constraint is not redundant, explain what configurations were feasible without it and become infeasible with it. If the constraint is logically entailed by the other constraints in the model, explain how.
\end{problem}

\begin{problem}
Suppose we have some discrete optimization or constraint satisfaction problem (not necessarily any of the ones we have seen in our course). How can you experimentally witness that some given constraint is redundant for the model? I.e. if you have your model coded in MiniZinc and can run Gecode on it or any its modification, what can you do to assure that a constraint does not make the set of all feasible solutions smaller?
\end{problem}

\begin{problem}
A \emph{Latin square} of order \(n\) is an \(n\times n\) matrix made out of the integers in \(\{1,2,\dots,n\}\) with the property that each of the these \(n\) integers occurs exactly once in each row and exactly once in each column of the array. For example
\[
\begin{matrix}
1& 2& 3& 4& 5\\

```latex
2& 3& 4& 5& 1\\
3& 4& 5& 1& 2\\
4& 5& 1& 2& 3\\
5& 1& 2& 3& 4
\end{matrix}\]
is a Latin square of order 5. Formulate the problem of finding a Latin square of order \(n\) in MiniZinc. Employ the \verb©alldifferent© constraint in your model.
\end{problem}

\begin{problem}
A \emph{Magic square} of order \(n\) is an \(n\times n\) matrix filled with integers \(\{1,2,\dots,n^2\}\) arranged in such a way that the sum of every row, column, and the two main diagonals is the same. Every integer is used only once in the matrix. For example
\[
\begin{matrix}
1 &15& 24& 8& 17\\
23& 7& 16& 5& 14\\
20& 4 &13& 22 &6\\
12& 21 &10 &19& 3\\
9 &18& 2& 11 &25
\end{matrix}\]
is a magic square of order 5, because each row, column and main diagonal sums up to 65. Formulate the problem of finding a magic square of order \(n\) in MiniZinc. Employ the \verb©alldifferent© constraint in your model.
\end{problem}


\begin{problem}\label{chess-pieces}
Consider an \(n\times n\) chess board. Suppose we have Rooks and Bishops (simultaneously on the same board) and we want to place them on the board in a way that no two pieces can attack each other. Suppose we receive respectively 2 and 3 units of profit per each piece of Rook and Bishop that we place on board.\footnote{See any resource on the internet to learn how Rooks and Bishops move/attack in chess. Feel free to ask TA in case you need further clarifications on the problem statement.}
\begin{subproblems}
\item Find the maximum profit we can get for \(n = 10\).
\item Find the maximum \(n\) for which your code can give the result in less than 10 seconds.
% \item What \emph{symmetry breaking} constraint could you propose for this problem?
\end{subproblems}
\end{problem}

\begin{problem}
Which of the following mathematical statements is/are true and why?
\begin{subproblems}
\item ? For every \(n\) every Latin square of order \(n\) has the entries of its first row going in increasing order.
\item ? For every \(n\) if Latin squares of order \(n\) exist, then there exists a Latin square that has the entries of its first row going in increasing order.
\item ? For every \(N\) every solution to the N-Queens problem has the following property: the column index of the queen in the first row is less than the column index of the queen in the bottom row.
\item ? For every \(N\) if N-Queens problem has solutions, then there exists a solution that has the following property: the column index of the queen in the first row is less than the column index of the queen in the bottom row.
\end{subproblems}
\end{problem}
```