1. Consider the box placement problem: given a set of rectangular (2-dimensional) boxes and a rectangular floor, to find a way to place the boxes so that the floor is fully covered without overlaps. To simplify, boxes cannot be rotated.

The input data of a box placement instance consist of: \( x_{\text{floor}} \) and \( y_{\text{floor}} \), the horizontal and vertical dimensions of the floor; \( n \), the number of boxes; and a sequence of \( n \) pairs of numbers \( x_{\text{side}}_i \) and \( y_{\text{side}}_i \) representing the horizontal and vertical dimensions of the \( i \)-th box.

Assume that all numbers are integers, and that \( x_{\text{floor}} \cdot y_{\text{floor}} = \sum_{i=1}^{n} x_{\text{side}}_i \cdot y_{\text{side}}_i \).

(a) (3 pts.) Complete the following program in Comet for solving the box placement problem.

```comet
string file = System.getArgs()[System.argc()-1];
ifstream data(file);
int xfloor = data.getInt();
int yfloor = data.getInt();
int n = data.getInt();
range Box = 1..n;
int xside[Box];
int yside[Box];
forall (i in Box) {
    xside[i] = data.getInt();
    yside[i] = data.getInt();
}
import cotfd;
Solver<CP> cp();

var<CP>{int} x[i in Box](cp, 1 .. );
var<CP>{int} y[i in Box](cp, 1 .. );

solve<cp> {
   forall (i in Box, j in Box : i < j)

    }
using { label(x); label(y); }
```

Hint: Just express that boxes do not overlap: given two boxes \( i \) and \( j \), enforce that box \( i \) is on the left, on the right, below or above box \( j \). As \( x_{\text{floor}} \cdot y_{\text{floor}} = \sum_{i=1}^{n} x_{\text{side}}_i \cdot y_{\text{side}}_i \), if boxes do not overlap they must cover the floor.

(b) (3 pts.) Note that in any solution, for each vertical line, if we sum the vertical sides of the traversed boxes, we get \( y_{\text{floor}} \); and similarly for horizontal lines. Write these redundant constraints in Comet.
2. A Latin square is an \( n \times n \) grid of natural numbers between 1 and \( n \) such that every row and every column contains different numbers. We want to complete the following partially filled Latin square:

\[
\begin{array}{cccc}
1 & & & \\
2 & 1 & & \\
& 2 & 1 & \\
\end{array}
\]

Let \( x_{ij} \) denote the value in the cell at row \( i \), column \( j \). The goal of this exercise is to see how to propagate the information of the already filled cells.

(a) \((1 \text{ pt.})\) Consider the constraint \textit{all different}(\( x_{11} \), \( x_{21} \), \( x_{31} \), \( x_{41} \)) stating that the numbers in the first column are different, with domains \( D(x_{11}) = \{1\} \), \( D(x_{21}) = \{2\} \), \( D(x_{31}) = D(x_{41}) = \{1, 2, 3, 4\} \). Draw the corresponding value graph. Enforce arc consistency, draw the resulting graph and identify the edges to be removed.

(b) \((1 \text{ pt.})\) Consider the constraint \textit{all different}(\( x_{12} \), \( x_{22} \), \( x_{32} \), \( x_{42} \)) stating that the numbers in the second column are different, with domains \( D(x_{22}) = \{1\} \), \( D(x_{32}) = \{2\} \), \( D(x_{12}) = D(x_{42}) = \{1, 2, 3, 4\} \). Draw the corresponding value graph. Enforce arc consistency, draw the resulting graph and identify the edges to be removed.

(c) \((1 \text{ pt.})\) Consider the constraint \textit{all different}(\( x_{13} \), \( x_{23} \), \( x_{33} \), \( x_{43} \)) stating that the numbers in the third column are different, with domains \( D(x_{33}) = \{1\} \), \( D(x_{13}) = D(x_{23}) = D(x_{43}) = \{1, 2, 3, 4\} \). Draw the corresponding value graph. Enforce arc consistency, draw the resulting graph and identify the edges to be removed.

(d) \((1 \text{ pt.})\) Consider the constraint \textit{all different}(\( x_{41} \), \( x_{42} \), \( x_{43} \), \( x_{44} \)) stating that the numbers in the fourth row are different, with domains \( D(x_{41}) = D(x_{42}) = D(x_{43}) = D(x_{44}) = \{1, 2, 3, 4\} \). Draw the value graph after propagating the edges removed in (a), (b) and (c). Enforce arc consistency and draw the resulting graph. Does any variable become fixed?
1. The problem of \textit{All-SAT} consists in obtaining \textit{all} models of a given propositional formula $F$.

   (a) (2.5 pts.) Assume you can use a CDCL SAT solver as a black box. Explain how to solve the All-SAT problem with it.

   (b) (2.5 pts.) Assume you have access to the code of a CDCL SAT-solver. Explain which changes you would make to the SAT solver in order to get a more efficient tool for solving All-SAT than in (a). Which advantage(s) would this have in comparison to your solution in (a)?

2. (5 pts.) The problem of Max-SAT is defined as follows. Given:

   \begin{itemize}
   \item a set $H$ of propositional clauses (called \textit{hard} clauses),
   \item a set $S$ of propositional clauses $C_1, C_2, \ldots, C_m$ (called \textit{soft} clauses), and
   \item a set of positive natural numbers (called \textit{weights}) $\omega_1, \omega_2, \ldots, \omega_m$ corresponding to soft clauses,
   \end{itemize}

the problem consists in finding the model of the hard clauses that minimizes the sum of the weights of the falsified soft clauses. More formally, if $X = (x_1, \ldots, x_n)$ are the propositional variables, the goal is to find $\sigma : X \rightarrow \{0, 1\}$ such that:

   \begin{itemize}
   \item For each $C \in H$, it holds $\sigma(C) = 1$.
   \item If $\sigma' : X \rightarrow \{0, 1\}$ is such that for each $C \in H$ it holds $\sigma(C) = 1$, then:
   \begin{align*}
   \sum_{C_i \in S} w_i & \leq \sum_{C_i \in S} w_i \\
   \sigma(C_i) = 0 & \quad \sigma'(C_i) = 0
   \end{align*}
   \end{itemize}

Explain how to solve a given Max-SAT instance by transforming it into an instance of Integer Linear Programming. Justify informally the correctness of the transformation.