It is still the year 2137. You are consultants and a client organises space travel for tourist groups. The client company has a fleet of spacecrafts of various amounts of seats, and it knows, for each tourist group, the group size and the language all group members can speak. Assuming there are as many spacecrafts of each size as needed, the company needs to decide for each group which spacecraft to use, such that:

• there are enough seats in each spacecraft to accommodate all the groups that travel in it;
• no group is spread over several spacecrafts;
• at most two languages are spoken aboard each spacecraft; and
• the total number of unused seats during all the spaceflights is minimal.

There is no limit to the number of groups in a spacecraft, except as dictated by the seat and language constraints. There is no limit to the number of used spacecrafts of any particular size. The only objective function is the total number of unused seats: the spacecrafts the unused seats are located in and the number of used spacecrafts are irrelevant.

Here is a small feasible STP instance. There are two spacecraft sizes: 6 and 8 seats. There are four groups, named $A$, $B$, $C$, and $D$, which all speak a different language and have the following respective sizes: 2, 3, 3, and 5 persons. An optimal solution assigns groups $A$ and $D$ to a spacecraft of size 8, and groups $B$ and $C$ to a spacecraft of size 6: there is only one unused seat, in the 8-seater, and there is no solution with no unused seats. Assigning also groups $B$ and $C$ to a spacecraft of size 8 is sub-optimal as the number of unused seats goes up to three. Assigning groups $A$, $B$, and $C$ to a spacecraft of size 8 and group $D$ to a spacecraft of size 6 also yields one unused seat but is incorrect as three languages are spoken aboard the 8-seater.

The company currently has one instance of the problem, with 20 spacecraft sizes and 111 tourist groups speaking a total of 88 languages. As the company managers do not know the power of combinatorial optimisation, they think that you might not be able to solve this instance to optimality in reasonable time. Hence they suggest you also try your approach by only taking into account the first $n$ tourist groups, with $n$ ranging from 2 to 111, thus generating 110
instances. A skeleton MiniZinc model with preprocessing to compute all the parameters based on the value of \( n \), which is called \( \text{nGroups} \) in the model, and a datafile for the large instance are at [http://user.it.uu.se/~pierref/courses/M4CO/assignments/assignment3.zip](http://user.it.uu.se/~pierref/courses/M4CO/assignments/assignment3.zip).

**Hints:** A straightforward (and acceptable) way to model this problem is to have a size variable for each spacecraft. However, note that it is possible to model the problem without modelling the size of each spacecraft explicitly.

With a good model, this problem can be solved to optimality very efficiently even for the largest given instance. However, coming up with such a model may be rather difficult and you do not need to find such a model to get the maximum score on this assignment. Perform the following sequence of tasks:

A. Design and evaluate a model. **Hints:** Think hard about different problem viewpoints, useful constraint predicates, sub-problem pre-solving towards using extensional constraint predicates (such as `element`, `table`, and `regular`), as well as exploiting symmetries in the problem, your model, or an instance.

B. Pick a CP or LCG backend. Experimentally evaluate at least six combinations\(^1\) of a variable selection strategy with a value selection strategy and consistencies for the constraints of your model of Task A, so that it becomes more efficient than without those annotations.

For the evaluations, start from \( n = 2 \) and report the results for the chosen backends for all the considered technologies. Increase \( n \) by steps of 1 if your model does not perform well under all chosen backends, by steps of 10 if your model can solve all instances under at least one backend, and otherwise by an appropriate step size between 1 and 10. **Use any time-out of a few CPU minutes per instance in order to avoid too long solving times.** Note that the difficulty of instances increases with \( n \), with some exceptions, hence you can stop the experiments when all chosen backends time out for five consecutive instances.

You are strongly advised to script your experiments in a way that makes it easy, and ideally automatic, to include the results in your report. For your convenience, here are the optimal objective values for some small values of \( n \):

<table>
<thead>
<tr>
<th>( n )</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused seats</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Submission Instructions

All task answers, other than source code, must be in a single report in PDF format; all other formats are rejected. Furthermore:

- Identify the team members and state the team number inside the report.
- Address each task of each problem, using the numbering and the ordering in which they appear in the assignment statement.
- Take the instructions of the demo report at [http://user.it.uu.se/~pierref/courses/M4CO/demoReport](http://user.it.uu.se/~pierref/courses/M4CO/demoReport) as a strict guideline for the structure and content of a model description and evaluation in the report, as well as an indication of its expected quality.
- If at least one teammate has taken a course on CP, then for Assignments 2 and 3 show experiments with annotations for CP inference and CP search: self-study and apply the slides of **Topic 8: Inference and Search for CP and LCG** (also see Chapter 6 in the **MiniZinc Tutorial**).

\(^1\) At least three combinations, for solo teams.
- Write clear task answers, source code, and comments.
- Justify all task answers, except where explicitly not required.
- State any assumptions you make that are not in this document.
- Thoroughly proofread, spellcheck, and grammar-check your report.
- Upload all models.
- Match exactly the uppercase, lowercase, and layout conventions of any filenames and I/O texts imposed by the tasks, as we will process your source code automatically.
- Write a paragraph, which will not be graded, describing your experience with this assignment: which aspects were too difficult or too easy, which aspects were interesting or boring? This will help us improve the course in the coming years.
- Remember that when submitting you implicitly certify that your report and all its uploaded attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, that each teammate can individually explain any part starting from the moment of submitting your report, and that your report and attachments are not freely accessible on a public repository.

Only one of the teammates submits the solution files (one PDF report with answers to all the tasks, and all model files), without folder structure and without compression, via the Student Portal, whose clock may differ from yours, by the given hard deadline.

**Grading Rules**

If all tasks have been seriously attempted and all requested models exist, have the comments exemplified in the demo report, and produce correct outputs for some of our grading instance data in reasonable time on the Linux computers of the IT department, within the given time bounds (if any), then you get at least 1 point (of 5), otherwise you get 0 points and fail the Assignments part of the course. Furthermore:

- If your models pass most of our grading tests and your report is complete, then you get 4 or 5 points, depending also on the quality of the model comments and the report; you are not invited to the grading session.
- If your models fail many of our grading tests or your report is incomplete, then you get an initial mark of 1 or 2 points, depending also on the quality of the model comments and the report; you are invited to the grading session, where you can try and increase your initial mark by 1 point into your final mark.

However, if the assistant figures out a minor fix that is needed to make your model run as per our instructions above, then, instead of giving 0 points up front, the assistant may deduct 1 point at his discretion.

Considering that there are two or three help sessions for each assignment, you must get minimum 1 point (of 5) on this assignment until the end of its grading session, and minimum 8 points (of 15) over all three assignments in order to pass the Assignments part (3 credits) of the course.