Examination Timetabling using PLR

João Xavier and João Ribeiro

Faculdade de Engenharia da Universidade do Porto, Portugal
ei06116@fe.up.pt, ei06019@fe.up.pt
FEUP-PL, 3MIEIC3, Group 28

Abstract. This paper describes an Examination Timetabling Problem solver, developed using Logic Programming with Restrictions and applied to the problem introduced by the International Timetabling Competition 2007. Model and datasets based on real world instances were provided by the competition organization.

1 Introduction

Determining the perfect exam schedule is a common problem among universities and other institutions. The main objective in the development of timetabling algorithms is to produce the best solutions, possible to every student and considering the available resources.

The core of the problem consists of scheduling exams into a given number of periods and rooms while satisfying a list of hard constraints. A feasible solution must satisfy all hard constraints. Soft constraints are used to introduce a scoring system to compare the quality of different solutions.

The following sections of this paper contain detailed information about the problem itself and the methods used to solve it, as well as test results and concluding remarks.

2 Problem Description

According to the information available at the ITC 2007 webpage [1], the present problem model extends the one most commonly worked upon. The core of the problem consists of scheduling exams into a given number of periods and rooms while satisfying a list of hard constraints. A feasible solution must satisfy all hard constraints. Soft constraints are used to introduce a scoring system to compare the quality of different solutions.

The problem is developed upon the following premises [2]:

- For all entities, sequential numbering beginning with 0 is assumed;
- An examination session is made of a number of periods over a specified length of time. Number and length of Periods are provided;
- A set of exams that are to be scheduled into periods. Exam codes are not provided;

- A set of students enrolled on individual exams. Each student is enrolled on a number of exams. Students enrolled on an exam are considered to take that examination. For each exam, the set of enrolled students is provided;
- A set of rooms with individual capacities is provided;
- Hard Constraints which must be satisfied;
- Soft Constraints which contribute to a penalty if they are violated;
- Details including a weighting of particular soft constraints.

A feasible timetable is one in which all examinations have been assigned to a period and room and all the following hard constraints are satisfied:

- No student sits more than one examination at the same time;
- The capacity of individual rooms is not exceeded at any time throughout the examination session;
- Period lengths are not violated;
- Satisfaction of period related hard constraints e.g. Exam A after Exam B;
- Satisfaction of room related hard constraints e.g. Exam A must use Room 101.

The soft constraints can be outlined as follows;

- *Two exams in a row*
  The number of occurrences when students have to sit two exams in a row on the same day;
- *Two exams in a day*
  The number of occurrences when students have to sit two exams on the same day. This constraint only becomes important when there are more than two examination periods in the one day. This is further explained later when the evaluation is described;
- *Specified spread of examinations*
  The number of occurrences when students have to sit more than one exam in a time period specified by the institution. This is often used in an attempt to be as fair as possible to all students taking exams;
- *Mixed duration of examinations within individual periods*
  The number of occurrences of exams timetabled in rooms along with other exams of differing time duration;
- *Larger examinations appearing later in the timetable*
  The number of large exams appearing in the latter portion of the timetable. Both large and later portion are user defined;
- *Period related soft constraints*
  The number of times a period is used which has an associated penalty. This is multiplied by the actual penalty as different periods may have different associated weightings;
- *Room related soft constraints*
  The number of times a room is used which has an associated penalty. This is multiplied by the actual penalty as different rooms may have different associated weightings.

These can effectively be split into two groups i.e. those which are resource specific and those which can have a global setting. Period related and room related constraints are resource specific i.e. settings can be established for each period and each room. This allows control of how resources would be used in constructing a solution. Values for these can be found after the introduction of periods and rooms in the datasets. All other soft constraints can be set relative to each other. These are referred to as global Settings.

Soft constraints' weights must be set directly into the input file; this is known as building the "Institutional Model". This allows the user to change such settings in order to attain the best solution for their own purposes.

3 Data Files

Datasets regarding information on exams, available rooms and periods, hard and soft constraints was available at the ITC website [3].

Information is read from a text file which presents it in the following format (sequential numbering always begins with 0):

Number of Exams

[Exams:4]
120, 1, 2, 3, 4
120, 3, 5
(...)

First line indicates the number of exams to consider. Every line after this one contains data concerning each exam: the first number is the duration, in minutes, and is followed by a list of number codes of students attending it.

Number of Periods

[Periods:3]
15:01:2009, 09:30:00, 120, 0
15:01:2009, 15:30:00, 150, 0
(...)

First line indicates the number of periods available. Every line after this one contains data concerning each period: the date, in standard date format (dd:mm:yyyy); the time, in 24 hour format (hh:mm:ss); the duration of the period, in minutes; the penalty associated with its use, a positive integer.

Number of Rooms

[Rooms:3]
3, 0
4, 5
(...)

First line indicates the number of rooms available. Every line after this one contains data concerning each room: the capacity, a positive integer; the penalty associated with its use, also a positive integer.

Period Related Hard Constraints

[PeriodHardConstraints]

1, AFTER, 0

0, EXCLUSION, 2

1, EXAM_COINCIDENCE, 2

(...)

This list provides rules concerning exams that are to be met for a solution to be feasible. First line contains the tag [PeriodHardConstraint].

X, AFTER, Y - forces exam X to be scheduled later than exam Y;

X, EXCLUSION, Y - forces exams X and Y not to be scheduled at the same period;

X, EXAM_COINCIDENCE, Y - forces exams X and Y to be scheduled at the same period.

Room Related Hard Constraints

[RoomHardConstraints]

0, ROOM_EXCLUSIVE

(...)

This list provides rules concerning rooms that are to be met for a solution to be feasible. First line contains the tag [RoomHardConstraints]. Every following line indicates which exams must be scheduled in a room by itself.

Institutional Model Weightings

[InstitutionalWeightings]

TWOINAROW, 7

TWOINADAY, 5

PERIODSPREAD, 1

NONMIXEDDURATIONS, 10

FRONTLOAD, 2, 1, 5

This list provides the values given to global soft constraints. First line contains the tag [InstitutionalWeightings]. Every following line contains a soft constraint and the penalty associated.

4 Decision Variables

According to the number of the exams to be considered, a list of the same size is created. For each of its positions, a pair (Period-Room) is created, representing an exam.

It is guaranteed that no two exams are scheduled in the same room at the same time.

5 Restrictions

5.1 Number of exams

The number of exams is restricted at the beginning with the predicate *length/2* as being equal to the number of exams listed in the input file.

5.2 Domain of exams

The upper bound of domain is calculated by the following formula:

$$RoomsNumber * PeriodsNumber - 1 \tag{1}$$

which is then applied to the predicate *domain/3*, with the lower bound 0.

5.3 All exams are different

This constraint is easily implemented with the *all_different/1* predicate, which grants that all exams in the list will have different combinations of room and period.

5.4 Exams duration and capacity don't exceed the limits

Defined by the predicate *capDurCons/5*, it is ensured that no exam duration exceeds the period where it is insered. It also checks for the number of students that will attend an exam, forcing it to occur on a room where its capacity can hold everyone who is assigned to that exam.

5.5 No student has more than one exam at a given period

This restriction prevents the same student from being on more than one exam at a given period. It is applied by the predicate *studentCons/3*, which, for each exam, checks if any other exam has any student attending the current exam, thus applying the constraint in those situations.

5.6 Period Hard Constraints

All the constraints from this section make a comparison between two exams according to the period when they will take place. The predicate *pHardCons/3* is in charge of the following constraints:

AFTER. Assures that the first exam given will occur after the last.

EXCLUSION. Assures that the first exam given will *never* occur at the same period the last.

EXAM_COINCIDENCE. Assures that the first exam given will occur at the same period the last.

5.7 Room Hard Constraints

The only constraint from this section is ROOM_EXCLUSIVE, which is evaluated by *rHardCons/3* and works in a similar fashion as the previous constraint. In this case, the predicate sets that the exam passed by parameter must be the only to take place in the room that it currently occupies.

6 Evaluation Function

In order to compare different feasible solutions, a scoring system is implemented, based on the number of soft constraint violations. The lowest score indicates the best solution. Scores are calculated according to the following (based on the ITC website - for further information and examples consult the website [4]).

6.1 Exams in a Row

Counts the number of occurrences where two examinations are taken by students straight after one another. Once this has been established, the number of students involved in each occurance should be added and multiplied by the "two in a row" penalty set in the "Institutional Model Index". Note that two exams in a row are not counted overnight and that this should be calculated for each occurence where a student or students have two exams in a row within the same day.

6.2 Two Exams in a Day

In the case where there are three periods or more in a day, count the number of occurrences of students having two exams in a day which are not directly adjacent, and multiply this by the "two in a day" weighting provided within the "Institutional Model Index". Therefore, two exams in a day are considered as those which are not adjacent i.e. they have a free period between them. This is done to ensure a particular exam placing within a solution does not contribute twice to the overall penalty. For example if Exam A and Exam B were in adjacent periods in the same day the penalty would be counted as part of the 'Two exams in a row penalty'.

6.3 Period Spread

This constraint allows an organisation to "spread" an individual's examinations over a specified number of periods. This can be thought of an extension of the two constraints previously described. Within the "Institutional Model Index", a figure is provided relating to how many periods the solution should be "optimised" over. The higher this figure, potentially the better the spread of examinations for individual students.

6.4 Mixed Durations

This applies a penalty to a ROOM and PERIOD (not Exam) where there are mixed durations, such that:

For Each Period

For Each Room

Penalty = (number of different durations -1) * 'NONMIXEDDURATIONS' weighting

6.5 Larger Exams towards the beginning of the examination session

It is desirable that examinations with the largest numbers of students are timetabled at the beginning of the examination session. In order to take account of this the FRONTLOAD expression is introduced. Within the Intuitional Model Index' the FRONTLOAD expression has three parameters.

First parameter = number of largest exams. Largest exams are specified by class size

Second parameter = number of last periods to take into account

Third parameter = the penalty or weighting

6.6 Room Penalty

It is often the case that organisations want to keep certain room usage to a minimum. As with the 'Mixed Durations' component of the overall penalty, this part of the overall penalty should be calculated on a period by period basis. For each period, if a room used within the solution has an associated penalty, the penalty for that room for that period is calculated by multiplying the associated penalty by the number of times the room is used.

6.7 Period Penalty

It is often the case that organisations want to keep certain period usage to a minimum. As with the "Mixed Durations" and the "Room Penalty" components of the overall penalty, this part of the overall penalty should be calculated on a

period by period basis. For each period the penalty is calculated by multiplying the associated penalty by the number of times the exams timetabled within that period.

7 Solution Visualization

The program outputs all solutions, along with their score, to a file specified in the source code (default: solution.txt on the c:*exam_tt**directory*).

The file stream is opened in append mode each time a different, feasible solution is going to be printed. *outputList/4* is in charge of properly preparing the information so it can be easily understood, with the format:

$$[\mathrm{P}_0 - R_0, P_1 - R_1, P_2 - R_2, \ldots, P_{n-1} - R_{n-1}]Score : S$$

Where P is the period, ranging from 0 to NUMBEROFPERIODS-1 and R is the room, ranging from 0 to NUMBEROFROOMS-1 from the exam specified by the index. S is the score evaluated by the soft constraints.

8 Results

A couple of small test cases were created following the ITC 2007 standards. This was due to the exaggerated amount of time that would take to compute a single feasible solution from the test cases available on the website.

The created test cases and their outputs can be checked on the appendix, section B and C. The first one, smaller, with some restrictions, runs faster than the latter, which is a bit more complex, featuring additional exams and number of students.

9 Conclusions and Future Work

In conclusion, PLR is an invaluable tool for programming in logic, correctly returning solutions to many types of problems, ranging from scheduling to solving puzzles. Although accurate, proves itself a bit slow and hard to optimize.

Comparing with the ITC '07 standards, our timetabling program evaluates all soft constraints properly and matches all hard constraints required, although it takes a long time to output feasible solutions on bigger test inputs.

A future version of this program would be a proper ITC candidate, more efficient timewise and only printing the best solution, matching the output rules.

References

1. *International Timetabling Competition. 2007-8-14*
http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index_files/examproblemmodel.htm/

2. *International Timetabling Competition. 2007-9-20*
http://www.cs.qub.ac.uk/itc2007/examtrack/report/Exam_Track_TechReportv4.0.pdf/
3. *International Timetabling Competition. 2007-7-30*
http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index_files/examinstances.htm/
4. *International Timetabling Competition. 2007-7-30*
http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index_files/examevaluation.htm/

Appendix

A Source Code

```
%
%       Sicstus PROLOG
%       exam_tt.pl
%
%       Copyright 2008  Joao Cristovao Xavier <ei06116@fe.up.pt>
%                       Joao Pedro Ribeiro <ei06019@fe.up.pt>
%
%       This program is free software; you can redistribute it and/or modify
%       it under the terms of the GNU General Public License as published by
%       the Free Software Foundation; either version 2 of the License, or
%       (at your option) any later version.
%
%       This program is distributed in the hope that it will be useful,
%       but WITHOUT ANY WARRANTY; without even the implied warranty of
%       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%       GNU General Public License for more details.
%
%       You should have received a copy of the GNU General Public License
%       along with this program; if not, write to the Free Software
%       Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
%       MA 02110-1301, USA.
%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PREPROCESSOR INSTRUCTIONS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:- use_module(library(lists)).
:- use_module(library(clpfd)).
:- use_module(library(system)).
:- include('./test cases/file2.test').
```

```
%%%%%%%%%%%%%
% MAIN CODE %
%%%%%%%%%%%%%


%begin
begin:-
% VARIABLE INITIALIZATIONS %
working_directory(_, 'C:\\exam_tt\\'),
FILENAME = 'solution.txt',
open(FILENAME, write, _),

periods(P),
rooms(R),
exams(E),
phc(PHC),
rhc(RHC),
iw(IW),

length(P, MAXPERIODS),
length(R, MAXROOMS),
length(E, NEXAMS),
MAXEXAMS is MAXPERIODS * MAXROOMS - 1,
compute(AllExams, MAXPERIODS, MAXROOMS),

% CONSTRAINTS %
length(Exams, NEXAMS),% constraint 1: Number of exams
domain(Exams, 0, MAXEXAMS),% constraint 2: Domain of exams
all_different(Exams),% constraint 3: All exams are different
capDurCons(Exams, E, AllExams, P, R),% constraint 4: Exams duration and
 capacity don't exceed the limits
studentCons(Exams, E, AllExams),% constraint 5: No student has more
than one exam at a given period
rHardCons(RHC, Exams, AllExams),% constraint 7: Room Hard Constraints
(ROOM_EXCLUSIVE)
pHardCons(PHC, Exams, AllExams),% constraint 6: Period Hard Constraints
(AFTER, EXCLUSION, EXAM_COINCIDENCE)

% CALCULATION / OUTPUT %
labeling([], Exams),
score(Exams, E, AllExams, R, P, IW, Score),
open(FILENAME, append, FILE),
```

```
outputList(Exams, AllExams, FILE, Score),
close(FILE), fail.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SOFT CONSTRAINTS EVALUATION %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%score(+Exams, +E, +AllExams, +Rooms, +Periods, +IW, -ScoreRPIW)
score(Exams, E, AllExams, Rooms, Periods, IW, ScoreRPIW):-
Score is 0,
scoreRoom(Exams, AllExams, Rooms, Score, ScoreR),
scorePeriod(Exams, AllExams, Periods, ScoreR, ScoreRP),
scoreIW(Exams, E, AllExams, Periods, Rooms, IW, ScoreRP, ScoreRPIW), !.

%scoreRoom(+Exams, +AllExams, +Rooms, +Score, -ScoreR)
scoreRoom([], _, _, Score, Score).
scoreRoom([H|T], AllExams, Rooms, Score, ScoreR):-
nth0(H, AllExams, _-R),
nth0(R, Rooms, Room),
getRoomPenalty(Room, Penalty),
ScoreTemp is Score + Penalty,
scoreRoom(T, AllExams, Rooms, ScoreTemp, ScoreR).

%scorePeriod(+Exams, +AllExams, +Periods, +Score, -ScoreP)
scorePeriod([], _, _, Score, Score).
scorePeriod([H|T], AllExams, Periods, Score, ScoreP):-
nth0(H, AllExams, P-_),
nth0(P, Periods, Period),
getPeriodPenalty(Period, Penalty),
ScoreTemp is Score + Penalty,
scorePeriod(T, AllExams, Periods, ScoreTemp, ScoreP).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INSTITUTIONAL WEIGHTINGS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%scoreIW(+Exams, +E, +AllExams, +Periods, +Rooms, +IW, +Score,
-ScoreIW)
scoreIW(_, _, _, _, _, [], Score, Score).
```

```
scoreIW(Exams, E, AllExams, Periods, Rooms, [[Constraint|R]|T],
 Score, ScoreIW):-
scoreIWAux(Constraint, Exams, E, AllExams, Periods, Rooms,
[Constraint|R], Score, ScoreResult),
scoreIW(Exams, E, AllExams, Periods, Rooms, T, ScoreResult,
ScoreIW).

%scoreIWAux(+Constraint, +Exams, +E, +AllExams, +Periods, +Rooms,
 +IW, +Score, -ScoreIW)
scoreIWAux('TWOINAROW', Exams, E, AllExams, Periods, _, IW, Score,
 ScoreIW):-
length(Periods, MAXPERIODS),
twoInARow(Exams, E, AllExams, MAXPERIODS, Periods, IW, Score,
 ScoreIW).

scoreIWAux('TWOINADAY', Exams, E, AllExams, Periods, _, IW, Score,
 ScoreIW):-
length(Periods, MAXPERIODS),
twoInADay(Exams, E, AllExams, MAXPERIODS, Periods, IW, Score,
 ScoreIW).

scoreIWAux('PERIODSPREAD', Exams, E, AllExams, _, _, IW, Score,
 ScoreIW):-
periodSpread(Exams, E, AllExams, IW, Score, ScoreIW).

scoreIWAux('NONMIXEDDURATIONS', Exams, _, AllExams, Periods,
Rooms, [_, Penalty], Score, ScoreIW):-
length(Rooms, RoomsLength),
length(RoomsPeriods, RoomsLength),
nonMixedDurations(0, RoomsPeriods, Periods, Exams, AllExams,
 Penalty, Score, ScoreIW).

scoreIWAux('FRONTLOAD', Exams, E, AllExams, Periods, _, [_,
LargerExams, LastP, Penalty], Score, ScoreIW):-
length(Periods, MAXPERIODS),
computeKeyList(Exams, E, KeyList),
keysort(KeyList, ReversedList),
reverse(ReversedList, SortedExamList),
TargetPeriod is MAXPERIODS - LastP,
frontLoad(SortedExamList, AllExams, 0, LargerExams, TargetPeriod,
 Penalty, Score, ScoreIW).



%%%%%%%%%%%%
```

```
% TWOINAROW %
%%%%%%%%%%%%%


%twoInARow(+Exams, +E, +AllExams, +MAXPERIODS, +Periods, +Score,
 -ScoreIW)
twoInARow([], _, _, _, _, _, Score, Score).
twoInARow([H|T], [EH|ET], AllExams, MAXPERIODS, Periods, IW,
Score, ScoreIW):-
twoInARowAux(H, T, EH, ET, AllExams, MAXPERIODS, Periods, IW,
 Score, ScoreTIAR),
twoInARow(T, ET, AllExams, MAXPERIODS, Periods, IW, ScoreTIAR,
 ScoreIW).

%twoInARowAux(+H, +T, +EH, +ET, +AllExams, +MAXPERIODS, +Periods,
 +IW, +Score,
-ScoreTIAR)
twoInARowAux(_, [], _, [], _, _, _, _, Score, Score).
twoInARowAux(H1, [H2|T], EH1, [EH2|ET], AllExams, MAXPERIODS,
 Periods, IW, Score,
ScoreTIAR):-
nth0(H1, AllExams, P1-_),
nth0(H2, AllExams, P2-_),
nth0(P1, Periods, Period1),
nth0(P2, Periods, Period2),
getPeriodDate(Period1, Date1),
getPeriodDate(Period2, Date2),
getStudentList(EH1, List1),
getStudentList(EH2, List2),
append(List1, List2, AllStudents),
length(AllStudents, AllStudentsLength),
remove_duplicates(AllStudents, NoDuplicates),
length(NoDuplicates, NoDuplicatesLength),
atom_codes(Date1, D1),
atom_codes(Date2, D2),
compareLists(D1, D2, EqualList),
sum_list(EqualList, SumList),
P1inc is P1 + 1,
IW = [_, Penalty],
(SumList #= 0 #/\ P1inc #= P2 #/\ AllStudentsLength
#\= NoDuplicatesLength)
#<=> TwoRow,
evaluateT(TwoRow, Score, Penalty, Result),
twoInARowAux(H1, T, EH1, ET, AllExams, MAXPERIODS, Periods, IW,
Result, ScoreTIAR).
```

```
%%%%%%%%%%%%%
% TWOINADAY %
%%%%%%%%%%%%%


%twoInADay(+Exams, +E, +AllExams, +MAXPERIODS, +Periods, +Score,
-ScoreIW)
twoInADay([], _, _, _, _, _, Score, Score).
twoInADay([H|T], [EH|ET], AllExams, MAXPERIODS, Periods, IW,
Score, ScoreIW):-
twoInADayAux(H, T, EH, ET, AllExams, MAXPERIODS, Periods, IW,
 Score, ScoreTIAD),
twoInADay(T, ET, AllExams, MAXPERIODS, Periods, IW, ScoreTIAD,
 ScoreIW).

%twoInADayAux(+H, +T, +EH, +ET, +AllExams, +MAXPERIODS, +Periods,
 +IW, +Score,
-ScoreTIAD)
twoInADayAux(_, [], _, [], _, _, _, _, Score, Score).
twoInADayAux(H1, [H2|T], EH1, [EH2|ET], AllExams, MAXPERIODS,
Periods, IW, Score,
 ScoreTIAD):-
nth0(H1, AllExams, P1-_),
nth0(H2, AllExams, P2-_),
nth0(P1, Periods, Period1),
nth0(P2, Periods, Period2),
getPeriodDate(Period1, Date1),
getPeriodDate(Period2, Date2),
getStudentList(EH1, List1),
getStudentList(EH2, List2),
append(List1, List2, AllStudents),
length(AllStudents, AllStudentsLength),
remove_duplicates(AllStudents, NoDuplicates),
length(NoDuplicates, NoDuplicatesLength),
atom_codes(Date1, D1),
atom_codes(Date2, D2),
compareLists(D1, D2, EqualList),
sum_list(EqualList, SumList),
P1inc is P1 + 1,
IW = [_, Penalty],
(SumList #= 0 #/\ P1inc #< P2 #/\ AllStudentsLength
#\= NoDuplicatesLength)
```

```
 #<=> TwoDay,
evaluateT(TwoDay, Score, Penalty, Result),
twoInADayAux(H1, T, EH1, ET, AllExams, MAXPERIODS, Periods,
 IW, Result,
ScoreTIAD).



%%%%%%%%%%%%%%%%
% PERIODSPREAD %
%%%%%%%%%%%%%%%%


%periodSpread(+Exams, +E, +AllExams, +IW, +Score, -ScoreIW)
periodSpread([], _, _, _, Score, Score).
periodSpread([H|T], [EH|ET], AllExams, IW, Score, ScoreIW):-
periodSpreadAux(H, T, EH, ET, AllExams, IW, Score, ScorePS),
periodSpread(T, ET, AllExams, IW, ScorePS, ScoreIW).

%periodSpreadAux(+H, +T, +EH, +ET, +AllExams, +IW, +Score,
 -ScorePS)
periodSpreadAux(_, [], _, [], _, _, Score, Score).
periodSpreadAux(H1, [H2|T], EH1, [EH2|ET], AllExams, IW, Score,
 ScorePS):-
nth0(H1, AllExams, P1-_),
nth0(H2, AllExams, P2-_),
getStudentList(EH1, List1),
getStudentList(EH2, List2),
append(List1, List2, AllStudents),
length(AllStudents, AllStudentsLength),
remove_duplicates(AllStudents, NoDuplicates),
length(NoDuplicates, NoDuplicatesLength),
IW = [_, Spread],
P1inc is P1 + Spread,
Penalty is AllStudentsLength - NoDuplicatesLength,
(P1inc #< P2 #/\ AllStudentsLength #\= NoDuplicatesLength)
 #<=> PSpread,
evaluateT(PSpread, Score, Penalty, Result),
periodSpreadAux(H1, T, EH1, ET, AllExams, IW, Result, ScorePS).



%%%%%%%%%%%%%%%%%%%%%%
% NONMIXEDDURATIONS %
%%%%%%%%%%%%%%%%%%%%%%
```

```
%nonMixedDurations(+Idx,  +RoomsPeriods, +Periods, +Exams,
+AllExams, +Penalty, +Score, -ScoreNMD)
nonMixedDurations(_, [], _, _, _, _, Score, Score).
nonMixedDurations(Idx, [H|T], Periods, Exams, AllExams, Penalty,
 Score, ScoreNMD):-
nonMixedDurationsAux(Idx, H, Periods, Exams, AllExams,
DiffDurations),
(DiffDurations #= 0) #<=> Value,
evaluateNMD(Value, Score, DiffDurations, Penalty, NMD),
IdxTemp is Idx+1,
nonMixedDurations(IdxTemp, T, Periods, Exams, AllExams, Penalty,
 NMD, ScoreNMD).

%nonMixedDurationsAux(+Idx, +RoomPeriod, +Periods, +Exams,
+AllExams, -DiffDurations)
nonMixedDurationsAux(_, RoomPeriod, _, [], _, DiffDurations):-
remove_duplicates(RoomPeriod, NoDuplicates),
length(NoDuplicates, DiffDurations).
nonMixedDurationsAux(Idx, RoomPeriod, Periods, [H|T], AllExams,
 DiffDurations):-
nth0(H, AllExams, P-R),
nth0(P, Periods, Period),
getPeriodDuration(Period, Duration),
(R #= Idx) #<=> Value,
evaluateRIdx(Value, RoomPeriod, Duration, Result),
nonMixedDurationsAux(Idx, Result, Periods, T, AllExams,
DiffDurations).



%%%%%%%%%%%%%
% FRONTLOAD %
%%%%%%%%%%%%%


%frontLoad(+SortedExamList, +AllExams, +Idx, +LargerExams,
+TargetPeriod, +Penalty, +Score, -ScoreFL)
frontLoad(_, _, LargerExams, LargerExams, _, _, Score, Score).
frontLoad([], _, _, _, _, _, Score, Score).
frontLoad([_-H|T], AllExams, Idx, LargerExams, TargetPeriod,
Penalty, Score, ScoreFL):-
nth0(H, AllExams, P-_),
(P #>= TargetPeriod) #<=> FL,
```

```
evaluateT(FL, Score, Penalty, Result),
IdxTemp is Idx + 1,
frontLoad(T, AllExams, IdxTemp, LargerExams, TargetPeriod,
 Penalty, Result, ScoreFL).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAPACITY / DURATION CONSTRAINT (C.4) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%capDurCons(+Exams, +E, +AllExams, +P, +R)
capDurCons([], [], _, _, _).
capDurCons([Idx|Exams], [EH|ET], AllExams, Periods, Rooms):-
getExamDuration(EH, Dur),
getExamNOS(EH, NOS),
nth0(Idx, AllExams, P-R),
nth0(P, Periods, D),
nth0(R, Rooms, C),
getPeriodDuration(D, Duration),
getRoomCapacity(C, Capacity),
Capacity #>= NOS,
Duration #>= Dur,
capDurCons(Exams, ET, AllExams, Periods, Rooms).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STUDENT CONSTRAINT (C.5) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%studentCons(+Exams, +E, +AllExams)
studentCons([], [], _).
studentCons([H|T], [EH|ET], AllExams):-
getStudentList(EH, List),
studentConsAux(H, List, T, ET, AllExams),
studentCons(T, ET, AllExams).

%studentConsAux(+E1, +List1, +Exams2, +E, +AllExams)
studentConsAux(_, _, [], [], _).
studentConsAux(E1, List1, [E2|T2], [EH|ET], AllExams):-
getStudentList(EH, List2),
append(List1, List2, AllStudents),
```

```
length(AllStudents, AllStudentsLength),
remove_duplicates(AllStudents, NoDuplicates),
length(NoDuplicates, NoDuplicatesLength),
(AllStudentsLength #= NoDuplicatesLength) #<=> Valid,
studentConstraint(Valid, E1, E2, AllExams),
studentConsAux(E1, List1, T2, ET, AllExams).

%studentConstraint(+Valid, +E1, +E2, +AllExams)
studentConstraint(1, _, _, _).
studentConstraint(0, E1, E2, AllExams):-
nth0(E1, AllExams, P1-_),
nth0(E2, AllExams, P2-_),
P1 #\= P2.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PERIOD HARD CONSTRAINTS (C.6) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%pHardCons(+PHC, +Exams, +AllExams)
pHardCons([], _, _).
pHardCons([[Idx1,Constraint,Idx2]|T], Exams, AllExams):-
nth0(Idx1, Exams, PR1),
nth0(Idx2, Exams, PR2),
nth0(PR1, AllExams, P1-_),
nth0(PR2, AllExams, P2-_),
periodConstraint(P1, P2, Constraint),
pHardCons(T, Exams, AllExams).

%periodConstraint(+P1, +P2, +Constraint)
periodConstraint(P1, P2, 'EXAM_COINCIDENCE'):-  P1 #= P2.
periodConstraint(P1, P2, 'EXCLUSION'):-  P1 #\= P2.
periodConstraint(P1, P2, 'AFTER'):-  P1 #> P2.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ROOM HARD CONSTRAINTS (C.7) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%rHardCons(+RHC, +Exams, +AllExams)
rHardCons([], _, _).
```

```
rHardCons([[Idx,Constraint]|T], Exams, AllExams):-
    roomConstraint(Idx, Constraint, Exams, AllExams),
rHardCons(T, Exams, AllExams).

%roomConstraint(+Idx, +Constraint, +Exams, +AllExams)
roomConstraint(Idx, 'ROOM_EXCLUSIVE', Exams, AllExams):-
nth0(Idx, Exams, PR),
nth0(PR, AllExams, _-Match),
roomExclusive(Match, Exams, AllExams, CountList),
sum(CountList, #=, 1).

%roomExclusive(+Match,, +Exams, +AllExams, -CountList)
roomExclusive(_, [], _, []).
roomExclusive(Match, [PR|ET], AllExams, [Value|T]):-
nth0(PR, AllExams, _-R),
(R #= Match) #<=> Value,
roomExclusive(Match, ET, AllExams, T).



%%%%%%%%%
% OUTPUT %
%%%%%%%%%


%outputList(+Exams, +AllExams, +Stream, +Score)
outputList(Exams, AllExams, FILE, Score):-
write(FILE, '['),
outputListAux(Exams, AllExams, FILE, Score).

%outputListAux(+Exams, +AllExams, +Stream, +Score)
outputListAux([H|[]], AllExams, FILE, Score):-
nth0(H, AllExams, P-R),
write(FILE, P-R), write(FILE, ']  '),
write(FILE, 'Score: '), write(FILE, Score),
nl(FILE), !.
outputListAux([H|T], AllExams, FILE, Score):-
nth0(H, AllExams, P-R),
write(FILE, P-R), write(FILE, ', '),
outputListAux(T, AllExams, FILE, Score).



%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALLEXAMS COMPUTATION %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%

%compute(-Exams, +UpX, +UpY)
compute(Exams, UpX, UpY):-
computeList([], 0, UpX, UpY, Exams), !.

%computeList(+Exams, +LowX, +Upx, +UpY, -Result)
computeList(Exams, UpX, UpX, _, Exams).
computeList(Exams, LowX, UpX, UpY, Temp):-
    computeListAux(ExamsTemp, LowX, 0, UpY),
append(Exams, ExamsTemp, Result),
    LowTempX is LowX+1,
    computeList(Result, LowTempX, UpX, UpY, Temp).

%computeListAux(-Exams, +LowX, +LowY, +UpY)
computeListAux([], _, UpY, UpY).
computeListAux([LowX-LowY|T], LowX, LowY, UpY):-
    LowTempY is LowY+1,
    computeListAux(T, LowX, LowTempY, UpY).



%%%%%%%%%%%%%%%%%%%%%%%%%
% AUXILIAR PREDICATES %
%%%%%%%%%%%%%%%%%%%%%%%%%


%evaluateT(+Value, +Score, +Penalty, -Result)
evaluateT(0, Score, _, Score).
evaluateT(1, Score, Penalty, Result):-
Result is Score + Penalty.

%evaluateNMD(+Value, +Score, +DiffDurations, +Penalty, -NMD)
evaluateNMD(0, Score, DiffDurations, Penalty, NMD):-
NMD is Score + (DiffDurations - 1)*Penalty.
evaluateNMD(1, Score, DiffDurations, _, NMD):-
NMD is Score + DiffDurations.

%evaluateRIdx(+Value, +RoomPeriod, +Duration, -Result)
evaluateRIdx(0, RoomPeriod, _, RoomPeriod).
evaluateRIdx(1, RoomPeriod, Duration, Result):-
append(RoomPeriod, [Duration], Result).

%compareLists(+List1, +List2, -EqualList)
```

```
compareLists([], [], []).
compareLists([], _, [1]).
compareLists(_, [], [1]).
compareLists([H1|T1], [H2|T2], [EH|ET]):-
(H1 #\= H2) #<=> EH,
compareLists(T1, T2, ET).

%computeKeyList(+Exams, +E, -Sorted)
computeKeyList([], [], []).
computeKeyList([H|T], [EH|ET], [NOS-H|ST]):-
getExamNOS(EH, NOS),
computeKeyList(T, ET, ST).



%%%%%%%%%%%%%%%%%%
% LIST UTILITIES %
%%%%%%%%%%%%%%%%%%


%getExamDuration(+Exam, -EH)
getExamDuration([EH|_], EH).

%getStudentList(+Exam, -List)
getStudentList([_|List], List).

%getExamNOS(+Exam, -NOS)
getExamNOS([_|ET], NOS):-
length(ET, NOS).

%getPeriodDate(+Period, -Date)
getPeriodDate([Date|_], Date).

%getPeriodDuration(+Period, -Duration)
getPeriodDuration(Period, Duration):-
nth0(2, Period, Duration).

%getPeriodPenalty(+Period, -Penalty)
getPeriodPenalty(Period, Penalty):-
nth0(3, Period, Penalty).

%getRoomCapacity(+Room, -Capacity)
getRoomCapacity([RC|_], RC).

%getRoomPenalty(+Room, -Penalty)
```

```
getRoomPenalty([_, RP], RP).
```

B Test Cases

B.1

```
[Exams:4]
120, 1, 2, 3, 4
120, 3, 5
150, 2, 6
180, 4, 7, 8
[Periods:3]
15:01:2009, 09:30:00, 120, 0
15:01:2009, 15:30:00, 150, 0
16:01:2009, 09:30:00, 180, 10
[Rooms:3]
3, 0
4, 5
2, 0
[PeriodHardConstraints]
1, AFTER, 0
0, EXCLUSION, 2
0, EXCLUSION, 1
1, EXAM_COINCIDENCE, 2
[RoomHardConstraints]
0, ROOM_EXCLUSIVE
[InstitutionalWeightings]
TWOINAROW, 7
TWOINADAY, 5
PERIODSPREAD, 1
NONMIXEDDURATIONS, 10
FRONTLOAD, 2, 1, 5
```

B.2

```
[Exams:6]
180, 4873
180, 4873, 7654
90, 97, 378, 454, 610, 1250, 1453, 1633, 2091, 2179, 2182, 2204, 2225, 2260, 2270,
 2300, 2415, 2504, 2516, 2527, 2581, 2590, 2609, 2633, 2673, 2686, 2698, 2762, 2763,
 2809, 2818, 2870, 2871, 2885, 2897, 2945, 3007, 3024, 3039, 3061, 3089, 3094, 3178,
 3208, 3210, 3247, 3314, 3315, 3319, 3345, 3373, 3409, 3411, 3464, 3504, 3550, 3570,
 3622, 3785, 3929, 3969, 3995, 4017, 4057, 4067, 4091, 4095, 4112, 4191, 5099, 5639,
 5641, 6001, 6917, 7581
```

```
90, 97, 378, 454, 610, 853, 1222, 1250, 1453, 1633, 2091, 2179, 2182, 2204, 2225,
 2260, 2270, 2300, 2415, 2504, 2516, 2527, 2581, 2590, 2609, 2633, 2673, 2686, 2698,
 2762, 2763, 2809, 2818, 2870, 2871, 2885, 2897, 2945, 3007, 3024, 3039, 3061, 3089,
 3094, 3178, 3208, 3210, 3247, 3314, 3315, 3319, 3345, 3373, 3409, 3411, 3464, 3504,
 3550, 3570, 3622, 3785, 3929, 3969, 3995, 4017, 4057, 4067, 4091, 4095, 4112, 4191,
 5099, 5639, 5641, 6001, 6917, 7581
120, 100, 6653, 6654, 7215, 7354, 7533, 7558, 7708, 7805, 7850, 7867
120, 100, 6653, 6654, 7215, 7354, 7533, 7558, 7708, 7805, 7850, 7867
[Periods:4]
12:01:2005, 09:30:00, 120, 0
12:01:2005, 14:30:00, 180, 0
13:01:2005, 09:30:00, 180, 0
13:01:2005, 14:30:00, 180, 0
[Rooms:3]
240, 0
210, 50
210, 0
[PeriodHardConstraints]
1, EXAM_COINCIDENCE, 3
2, EXAM_COINCIDENCE, 5
4, AFTER, 0,
5, EXCLUSION, 4
[RoomHardConstraints]
2, ROOM_EXCLUSIVE
[InstitutionalWeightings]
TWOINAROW, 20
TWOINADAY, 5
PERIODSPREAD, 20
NONMIXEDDURATIONS, 2
FRONTLOAD, 2, 1, 10
```

C Test Case Results

C.1

```
[0-1, 1-0, 1-2, 2-0]  Score: 45
[0-1, 1-2, 1-0, 2-0]  Score: 45
```

C.2

```
[1-0, 2-0, 0-1, 2-2, 3-0, 0-0]  Score: 52
[1-0, 2-0, 0-1, 2-2, 3-0, 0-2]  Score: 52
[1-0, 2-0, 0-1, 2-2, 3-2, 0-0]  Score: 52
[1-0, 2-0, 0-1, 2-2, 3-2, 0-2]  Score: 52
[1-0, 2-0, 0-2, 2-1, 3-0, 0-0]  Score: 52
```

[1-0, 2-0, 0-2, 2-1, 3-0, 0-1] Score: 102
[1-0, 2-0, 0-2, 2-1, 3-1, 0-0] Score: 102
[1-0, 2-0, 0-2, 2-1, 3-1, 0-1] Score: 152
[1-0, 2-0, 1-1, 2-2, 3-0, 1-2] Score: 50
[1-0, 2-0, 1-1, 2-2, 3-2, 1-2] Score: 50
[1-0, 2-0, 1-2, 2-1, 3-0, 1-1] Score: 100
[1-0, 2-0, 1-2, 2-1, 3-1, 1-1] Score: 150
[1-0, 2-1, 0-2, 2-0, 3-0, 0-0] Score: 52
[1-0, 2-1, 0-2, 2-0, 3-0, 0-1] Score: 102
[1-0, 2-1, 0-2, 2-0, 3-1, 0-0] Score: 102
[1-0, 2-1, 0-2, 2-0, 3-1, 0-1] Score: 152
[1-0, 2-1, 1-2, 2-0, 3-0, 1-1] Score: 100
[1-0, 2-1, 1-2, 2-0, 3-1, 1-1] Score: 150
[1-0, 2-2, 0-1, 2-0, 3-0, 0-0] Score: 52
[1-0, 2-2, 0-1, 2-0, 3-0, 0-2] Score: 52
[1-0, 2-2, 0-1, 2-0, 3-2, 0-0] Score: 52
[1-0, 2-2, 0-1, 2-0, 3-2, 0-2] Score: 52
[1-0, 2-2, 1-1, 2-0, 3-0, 1-2] Score: 50
[1-0, 2-2, 1-1, 2-0, 3-2, 1-2] Score: 50
[1-0, 3-0, 0-1, 3-2, 2-0, 0-0] Score: 62
[1-0, 3-0, 0-1, 3-2, 2-0, 0-2] Score: 62
[1-0, 3-0, 0-1, 3-2, 2-2, 0-0] Score: 62
[1-0, 3-0, 0-1, 3-2, 2-2, 0-2] Score: 62
[1-0, 3-0, 0-2, 3-1, 2-0, 0-0] Score: 62
[1-0, 3-0, 0-2, 3-1, 2-0, 0-1] Score: 112
[1-0, 3-0, 0-2, 3-1, 2-1, 0-0] Score: 112
[1-0, 3-0, 0-2, 3-1, 2-1, 0-1] Score: 162
[1-0, 3-0, 1-1, 3-2, 2-0, 1-2] Score: 60
[1-0, 3-0, 1-1, 3-2, 2-2, 1-2] Score: 60
[1-0, 3-0, 1-2, 3-1, 2-0, 1-1] Score: 110
[1-0, 3-0, 1-2, 3-1, 2-1, 1-1] Score: 160
[1-0, 3-1, 0-2, 3-0, 2-0, 0-0] Score: 62
[1-0, 3-1, 0-2, 3-0, 2-0, 0-1] Score: 112
[1-0, 3-1, 0-2, 3-0, 2-1, 0-0] Score: 112
[1-0, 3-1, 0-2, 3-0, 2-1, 0-1] Score: 162
[1-0, 3-1, 1-2, 3-0, 2-0, 1-1] Score: 110
[1-0, 3-1, 1-2, 3-0, 2-1, 1-1] Score: 160
[1-0, 3-2, 0-1, 3-0, 2-0, 0-0] Score: 62
[1-0, 3-2, 0-1, 3-0, 2-0, 0-2] Score: 62
[1-0, 3-2, 0-1, 3-0, 2-2, 0-0] Score: 62
[1-0, 3-2, 0-1, 3-0, 2-2, 0-2] Score: 62
[1-0, 3-2, 1-1, 3-0, 2-0, 1-2] Score: 60
[1-0, 3-2, 1-1, 3-0, 2-2, 1-2] Score: 60
[1-1, 2-0, 0-2, 2-1, 3-0, 0-0] Score: 102
[1-1, 2-0, 0-2, 2-1, 3-0, 0-1] Score: 152

[1-1, 2-0, 0-2, 2-1, 3-1, 0-0] Score: 152
[1-1, 2-0, 0-2, 2-1, 3-1, 0-1] Score: 202
[1-1, 2-0, 1-2, 2-1, 3-0, 1-0] Score: 100
[1-1, 2-0, 1-2, 2-1, 3-1, 1-0] Score: 150
[1-1, 2-1, 0-0, 2-2, 3-1, 0-1] Score: 202
[1-1, 2-1, 0-0, 2-2, 3-1, 0-2] Score: 152
[1-1, 2-1, 0-0, 2-2, 3-2, 0-1] Score: 152
[1-1, 2-1, 0-0, 2-2, 3-2, 0-2] Score: 102
[1-1, 2-1, 0-2, 2-0, 3-0, 0-0] Score: 102
[1-1, 2-1, 0-2, 2-0, 3-0, 0-1] Score: 152
[1-1, 2-1, 0-2, 2-0, 3-1, 0-0] Score: 152
[1-1, 2-1, 0-2, 2-0, 3-1, 0-1] Score: 202
[1-1, 2-1, 1-0, 2-2, 3-1, 1-2] Score: 150
[1-1, 2-1, 1-0, 2-2, 3-2, 1-2] Score: 100
[1-1, 2-1, 1-2, 2-0, 3-0, 1-0] Score: 100
[1-1, 2-1, 1-2, 2-0, 3-1, 1-0] Score: 150
[1-1, 2-2, 0-0, 2-1, 3-1, 0-1] Score: 202
[1-1, 2-2, 0-0, 2-1, 3-1, 0-2] Score: 152
[1-1, 2-2, 0-0, 2-1, 3-2, 0-1] Score: 152
[1-1, 2-2, 0-0, 2-1, 3-2, 0-2] Score: 102
[1-1, 2-2, 1-0, 2-1, 3-1, 1-2] Score: 150
[1-1, 2-2, 1-0, 2-1, 3-2, 1-2] Score: 100
[1-1, 3-0, 0-2, 3-1, 2-0, 0-0] Score: 112
[1-1, 3-0, 0-2, 3-1, 2-0, 0-1] Score: 162
[1-1, 3-0, 0-2, 3-1, 2-1, 0-0] Score: 162
[1-1, 3-0, 0-2, 3-1, 2-1, 0-1] Score: 212
[1-1, 3-0, 1-2, 3-1, 2-0, 1-0] Score: 110
[1-1, 3-0, 1-2, 3-1, 2-1, 1-0] Score: 160
[1-1, 3-1, 0-0, 3-2, 2-1, 0-1] Score: 212
[1-1, 3-1, 0-0, 3-2, 2-1, 0-2] Score: 162
[1-1, 3-1, 0-0, 3-2, 2-2, 0-1] Score: 162
[1-1, 3-1, 0-0, 3-2, 2-2, 0-2] Score: 112
[1-1, 3-1, 0-2, 3-0, 2-0, 0-0] Score: 112
[1-1, 3-1, 0-2, 3-0, 2-0, 0-1] Score: 162
[1-1, 3-1, 0-2, 3-0, 2-1, 0-0] Score: 162
[1-1, 3-1, 0-2, 3-0, 2-1, 0-1] Score: 212
[1-1, 3-1, 1-0, 3-2, 2-1, 1-2] Score: 160
[1-1, 3-1, 1-0, 3-2, 2-2, 1-2] Score: 110
[1-1, 3-1, 1-2, 3-0, 2-0, 1-0] Score: 110
[1-1, 3-1, 1-2, 3-0, 2-1, 1-0] Score: 160
[1-1, 3-2, 0-0, 3-1, 2-1, 0-1] Score: 212
[1-1, 3-2, 0-0, 3-1, 2-1, 0-2] Score: 162
[1-1, 3-2, 0-0, 3-1, 2-2, 0-1] Score: 162
[1-1, 3-2, 0-0, 3-1, 2-2, 0-2] Score: 112
[1-1, 3-2, 1-0, 3-1, 2-1, 1-2] Score: 160

[1-1, 3-2, 1-0, 3-1, 2-2, 1-2] Score: 110
[1-2, 2-0, 0-1, 2-2, 3-0, 0-0] Score: 52
[1-2, 2-0, 0-1, 2-2, 3-0, 0-2] Score: 52
[1-2, 2-0, 0-1, 2-2, 3-2, 0-0] Score: 52
[1-2, 2-0, 0-1, 2-2, 3-2, 0-2] Score: 52
[1-2, 2-0, 1-1, 2-2, 3-0, 1-0] Score: 50
[1-2, 2-0, 1-1, 2-2, 3-2, 1-0] Score: 50
[1-2, 2-1, 0-0, 2-2, 3-1, 0-1] Score: 152
[1-2, 2-1, 0-0, 2-2, 3-1, 0-2] Score: 102
[1-2, 2-1, 0-0, 2-2, 3-2, 0-1] Score: 102
[1-2, 2-1, 0-0, 2-2, 3-2, 0-2] Score: 52
[1-2, 2-1, 1-0, 2-2, 3-1, 1-1] Score: 150
[1-2, 2-1, 1-0, 2-2, 3-2, 1-1] Score: 100
[1-2, 2-2, 0-0, 2-1, 3-1, 0-1] Score: 152
[1-2, 2-2, 0-0, 2-1, 3-1, 0-2] Score: 102
[1-2, 2-2, 0-0, 2-1, 3-2, 0-1] Score: 102
[1-2, 2-2, 0-0, 2-1, 3-2, 0-2] Score: 52
[1-2, 2-2, 0-1, 2-0, 3-0, 0-0] Score: 52
[1-2, 2-2, 0-1, 2-0, 3-0, 0-2] Score: 52
[1-2, 2-2, 0-1, 2-0, 3-2, 0-0] Score: 52
[1-2, 2-2, 0-1, 2-0, 3-2, 0-2] Score: 52
[1-2, 2-2, 1-0, 2-1, 3-1, 1-1] Score: 150
[1-2, 2-2, 1-0, 2-1, 3-2, 1-1] Score: 100
[1-2, 2-2, 1-1, 2-0, 3-0, 1-0] Score: 50
[1-2, 2-2, 1-1, 2-0, 3-2, 1-0] Score: 50
[1-2, 3-0, 0-1, 3-2, 2-0, 0-0] Score: 62
[1-2, 3-0, 0-1, 3-2, 2-0, 0-2] Score: 62
[1-2, 3-0, 0-1, 3-2, 2-2, 0-0] Score: 62
[1-2, 3-0, 0-1, 3-2, 2-2, 0-2] Score: 62
[1-2, 3-0, 1-1, 3-2, 2-0, 1-0] Score: 60
[1-2, 3-0, 1-1, 3-2, 2-2, 1-0] Score: 60
[1-2, 3-1, 0-0, 3-2, 2-1, 0-1] Score: 162
[1-2, 3-1, 0-0, 3-2, 2-1, 0-2] Score: 112
[1-2, 3-1, 0-0, 3-2, 2-2, 0-1] Score: 112
[1-2, 3-1, 0-0, 3-2, 2-2, 0-2] Score: 62
[1-2, 3-1, 1-0, 3-2, 2-1, 1-1] Score: 160
[1-2, 3-1, 1-0, 3-2, 2-2, 1-1] Score: 110
[1-2, 3-2, 0-0, 3-1, 2-1, 0-1] Score: 162
[1-2, 3-2, 0-0, 3-1, 2-1, 0-2] Score: 112
[1-2, 3-2, 0-0, 3-1, 2-2, 0-1] Score: 112
[1-2, 3-2, 0-0, 3-1, 2-2, 0-2] Score: 62
[1-2, 3-2, 0-1, 3-0, 2-0, 0-0] Score: 62
[1-2, 3-2, 0-1, 3-0, 2-0, 0-2] Score: 62
[1-2, 3-2, 0-1, 3-0, 2-2, 0-0] Score: 62
[1-2, 3-2, 0-1, 3-0, 2-2, 0-2] Score: 62

```
[1-2, 3-2, 1-0, 3-1, 2-1, 1-1]  Score: 160
[1-2, 3-2, 1-0, 3-1, 2-2, 1-1]  Score: 110
[1-2, 3-2, 1-1, 3-0, 2-0, 1-0]  Score: 60
[1-2, 3-2, 1-1, 3-0, 2-2, 1-0]  Score: 60
[2-0, 1-0, 0-1, 1-2, 3-0, 0-0]  Score: 72
[2-0, 1-0, 0-1, 1-2, 3-0, 0-2]  Score: 72
[2-0, 1-0, 0-1, 1-2, 3-2, 0-0]  Score: 72
[2-0, 1-0, 0-1, 1-2, 3-2, 0-2]  Score: 72
[2-0, 1-0, 0-2, 1-1, 3-0, 0-0]  Score: 72
[2-0, 1-0, 0-2, 1-1, 3-0, 0-1]  Score: 122
[2-0, 1-0, 0-2, 1-1, 3-1, 0-0]  Score: 122
[2-0, 1-0, 0-2, 1-1, 3-1, 0-1]  Score: 172
[2-0, 1-0, 2-1, 1-2, 3-0, 2-2]  Score: 50
[2-0, 1-0, 2-1, 1-2, 3-2, 2-2]  Score: 50
[2-0, 1-0, 2-2, 1-1, 3-0, 2-1]  Score: 100
[2-0, 1-0, 2-2, 1-1, 3-1, 2-1]  Score: 150
[2-0, 1-1, 0-2, 1-0, 3-0, 0-0]  Score: 72
[2-0, 1-1, 0-2, 1-0, 3-0, 0-1]  Score: 122
[2-0, 1-1, 0-2, 1-0, 3-1, 0-0]  Score: 122
[2-0, 1-1, 0-2, 1-0, 3-1, 0-1]  Score: 172
[2-0, 1-1, 2-2, 1-0, 3-0, 2-1]  Score: 100
[2-0, 1-1, 2-2, 1-0, 3-1, 2-1]  Score: 150
[2-0, 1-2, 0-1, 1-0, 3-0, 0-0]  Score: 72
[2-0, 1-2, 0-1, 1-0, 3-0, 0-2]  Score: 72
[2-0, 1-2, 0-1, 1-0, 3-2, 0-0]  Score: 72
[2-0, 1-2, 0-1, 1-0, 3-2, 0-2]  Score: 72
[2-0, 1-2, 2-1, 1-0, 3-0, 2-2]  Score: 50
[2-0, 1-2, 2-1, 1-0, 3-2, 2-2]  Score: 50
[2-1, 1-0, 0-2, 1-1, 3-0, 0-0]  Score: 122
[2-1, 1-0, 0-2, 1-1, 3-0, 0-1]  Score: 172
[2-1, 1-0, 0-2, 1-1, 3-1, 0-0]  Score: 172
[2-1, 1-0, 0-2, 1-1, 3-1, 0-1]  Score: 222
[2-1, 1-0, 2-2, 1-1, 3-0, 2-0]  Score: 100
[2-1, 1-0, 2-2, 1-1, 3-1, 2-0]  Score: 150
[2-1, 1-1, 0-0, 1-2, 3-1, 0-1]  Score: 222
[2-1, 1-1, 0-0, 1-2, 3-1, 0-2]  Score: 172
[2-1, 1-1, 0-0, 1-2, 3-2, 0-1]  Score: 172
[2-1, 1-1, 0-0, 1-2, 3-2, 0-2]  Score: 122
[2-1, 1-1, 0-2, 1-0, 3-0, 0-0]  Score: 122
[2-1, 1-1, 0-2, 1-0, 3-0, 0-1]  Score: 172
[2-1, 1-1, 0-2, 1-0, 3-1, 0-0]  Score: 172
[2-1, 1-1, 0-2, 1-0, 3-1, 0-1]  Score: 222
[2-1, 1-1, 2-0, 1-2, 3-1, 2-2]  Score: 150
[2-1, 1-1, 2-0, 1-2, 3-2, 2-2]  Score: 100
[2-1, 1-1, 2-2, 1-0, 3-0, 2-0]  Score: 100
```

```
[2-1, 1-1, 2-2, 1-0, 3-1, 2-0]  Score: 150
[2-1, 1-2, 0-0, 1-1, 3-1, 0-1]  Score: 222
[2-1, 1-2, 0-0, 1-1, 3-1, 0-2]  Score: 172
[2-1, 1-2, 0-0, 1-1, 3-2, 0-1]  Score: 172
[2-1, 1-2, 0-0, 1-1, 3-2, 0-2]  Score: 122
[2-1, 1-2, 2-0, 1-1, 3-1, 2-2]  Score: 150
[2-1, 1-2, 2-0, 1-1, 3-2, 2-2]  Score: 100
[2-2, 1-0, 0-1, 1-2, 3-0, 0-0]  Score: 72
[2-2, 1-0, 0-1, 1-2, 3-0, 0-2]  Score: 72
[2-2, 1-0, 0-1, 1-2, 3-2, 0-0]  Score: 72
[2-2, 1-0, 0-1, 1-2, 3-2, 0-2]  Score: 72
[2-2, 1-0, 2-1, 1-2, 3-0, 2-0]  Score: 50
[2-2, 1-0, 2-1, 1-2, 3-2, 2-0]  Score: 50
[2-2, 1-1, 0-0, 1-2, 3-1, 0-1]  Score: 172
[2-2, 1-1, 0-0, 1-2, 3-1, 0-2]  Score: 122
[2-2, 1-1, 0-0, 1-2, 3-2, 0-1]  Score: 122
[2-2, 1-1, 0-0, 1-2, 3-2, 0-2]  Score: 72
[2-2, 1-1, 2-0, 1-2, 3-1, 2-1]  Score: 150
[2-2, 1-1, 2-0, 1-2, 3-2, 2-1]  Score: 100
[2-2, 1-2, 0-0, 1-1, 3-1, 0-1]  Score: 172
[2-2, 1-2, 0-0, 1-1, 3-1, 0-2]  Score: 122
[2-2, 1-2, 0-0, 1-1, 3-2, 0-1]  Score: 122
[2-2, 1-2, 0-0, 1-1, 3-2, 0-2]  Score: 72
[2-2, 1-2, 0-1, 1-0, 3-0, 0-0]  Score: 72
[2-2, 1-2, 0-1, 1-0, 3-0, 0-2]  Score: 72
[2-2, 1-2, 0-1, 1-0, 3-2, 0-0]  Score: 72
[2-2, 1-2, 0-1, 1-0, 3-2, 0-2]  Score: 72
[2-2, 1-2, 2-0, 1-1, 3-1, 2-1]  Score: 150
[2-2, 1-2, 2-0, 1-1, 3-2, 2-1]  Score: 100
[2-2, 1-2, 2-1, 1-0, 3-0, 2-0]  Score: 50
[2-2, 1-2, 2-1, 1-0, 3-2, 2-0]  Score: 50
```