DTU Course 02156 Logical Systems and Logic Programming

Mandatory Assignment 2 — Deadline Sunday 8/10 23:59

MUST BE SOLVED INDIVIDUALLY

You are only allowed to get help from the teacher and the teaching assistants.

You may use the notes and definitions on the first page of the sample exams.

You are allowed to use your computer and there is no 2 hours time limit.

For the whole assignment you must submit exactly 2 files on DTU Learn:

1. A single PDF file with extension `.pdf` with the report.
2. A single Prolog file with extension `.pl` with the programs.

Absolutely no ZIP files, no text processing documents or any other file formats.

The report should not contain any program listings (just refer to the Prolog file).

The programs should load in SWI-Prolog without any errors or warnings.

The programs must be properly documented with comments.

If possible use only the ISO Prolog features of SWI-Prolog covered in the course.

All programs must be tested and the tests must be included in the report.

In particular:

- Test a few normal cases.
- Test the special cases.
- Test with variables only (if the instantiation pattern allows variables).

Show the Prolog queries and the corresponding answers — and keep explanations short.

Problem 1 (20%)

Consider the following fragment of a word frequency list for a large English text:

```
w(5,2186369,a,det).
w(2107,4249,abandon,v).
w(5204,1110,abbey,n).
w(966,10468,ability,n).
w(321,30454,able,a).
w(6277,809,abnormal,a).
w(3862,1744,abolish,v).
w(5085,1154,abolition,n).
w(4341,1471,abortion,n).
w(179,52561,about,adv).
w(69,144554,about,prep).
w(3341,2139,above,a).
w(942,10719,above,adv).
w(786,12889,above,prep).
w(2236,3941,abroad,adv).
w(5106,1146,abruptly,adv).
```

The format is: `w(SortOrder,Frequency,Word,WordClass)`

`SortOrder` is `1` for the most frequent word. `WordClass` is the category: `det` for a determiner, `v` for verb, `n` for a noun, `a` for adjective, and so on.

Question 1.1

Write a deterministic program `ambiguous(+Word)` that succeeds if and only if `Word` is in the word frequency list with more than one category (standard arithmetic predicates can be used).

Question 1.2

Write a deterministic program `display` that prints the n most frequent words and corresponding categories in the word frequency list, where n is an argument to the predicate, hence for the above fragment for example:

```
?- display(500).
a det
able a
about adv
about prep

Yes
```

A historical comment:

More information about the word frequency list — based on the British National Corpus (BNC) — is available here: `https://www.kilgarriff.co.uk/bnc-readme.html`

Problem 2 (20%)

Consider the following formula: $(p \wedge q) \rightarrow (q \wedge p)$

Question 2.1

Use refutation and the systematic construction of a semantic tableau.

State whether this shows that the formula is valid or not.

Question 2.2

Provide a proof in the Gentzen system $\mathcal{G}$ of the formula.

Question 2.3

Complete the following incomplete proof in the Hilbert system $\mathcal{H}$ of the formula:

1.	$\{q \rightarrow \neg p, \neg\neg p\} \vdash ...$	Assumption
2.	$\{q \rightarrow \neg p, \neg\neg p\} \vdash ...$	Contrapositive 1
3.	$\{q \rightarrow \neg p, \neg\neg p\} \vdash ...$	Assumption
4.	$\{q \rightarrow \neg p, \neg\neg p\} \vdash ...$	MP 2,3
5.	$\{q \rightarrow \neg p, \neg\neg p\} \vdash ...$	Double negation 4
6.	$\{q \rightarrow \neg p\} \vdash ...$	Deduction 5
7.	$\{q \rightarrow \neg p\} \vdash ...$	Contrapositive 6
8.	$\{q \rightarrow \neg p\} \vdash ...$	Contrapositive 7
9.	$\vdash ...$	Deduction 8
10.	$\vdash ...$	Contrapositive 9
11.	$\vdash (p \wedge q) \rightarrow (q \wedge p)$	Def. of $\wedge$

Hint: It is like a little puzzle.

Problem 3 (20%)

Consider a Prolog program `firstlast(?List)` that succeeds if and only if `List` is a list with at least two elements and the first element is the same as the last element.

Sample queries:

```
?- firstlast([1,2,3,1]).

Yes

?- firstlast([1,2,3]).

No
```

Question 3.1

Express the predicate `firstlast` not using any other predicate (recursion allowed).

Question 3.2

Express the predicate `firstlast` using the predicate `append` only (no recursion allowed). It is pretty simple. Call the defined predicate `firstlasta` (`firstlast` via `append`).

Problem 4 (20%)

Consider the following Prolog program serving as a database of students in a course and their scores in a test and in the exam (a score is an integer between 0 and 100):

```
score(test,xenia,50).
score(test,alice,99).
score(test,bruce,22).
score(test,carol,77).
score(test,dorit,50).
score(test,erica,22).
score(exam,peter,42).
score(exam,alice,11).
score(exam,bruce,88).
score(exam,carol,33).
score(exam,dorit,50).
score(exam,erica,66).
score(exam,james,77).
```

For example, `xenia` scored 50 in the test but did not participate in the exam, and `alice` scored 99 in the test but only 11 in the exam.

Question 4.1

Write a deterministic Prolog program `test(+Integer)` that succeeds if and only if `Integer` is a score of a student who participated in the test or in the exam (or both).

Sample queries:

```
?- test(11).

Yes


?- test(12).

No


?- test(77).

Yes
```

Question 4.2

Write a deterministic Prolog program `problem` that prints the students who's score in the test was more than twice the score in the exam.

A sample query:

```
?- problem.

alice
carol

Yes
```

Problem 5 (20%)

Use the Isabelle sequent calculus formalization to prove the formula $((p \to q) \to p) \to p$ like in the `Logic_Examples.thy` file.

Insert the Isabelle lines in the Prolog file as a comment.

Sample lines for the formula $p \to p$:

```
proposition ‹[] ≫ (p → p) # []›
proof -
  from Imp_R have ?thesis if ‹
    p # []
    ≫
    p # []
    ›
    using that by force
  with Basic show ?thesis
    by force
qed
```