cps721: Assignment 1 (100 points).
Due date: Electronic file - Monday, September 28, 2020, before 21:00. Read Page 5.

YOU SHOULD NOT USE ";" , "!" AND "–>" IN YOUR PROLOG RULES.

You MUST work in groups of TWO, or THREE, you cannot work alone. You can discuss this assignment only with your CPS721 group partners or with the CPS721 instructor. By submitting this assignment you acknowledge that you read and understood the course Policy on Collaboration in homework assignments stated in the CPS721 course management form.

1 (40 points). This problem is designed to get you started in Prolog. More specifically, you have to do the following. Make up a simple knowledge base (KB) of 10-15 atomic sentences about bank accounts (not real ones), so that most of the queries below will successfully retrieve answers from your KB. Use the following predicates.

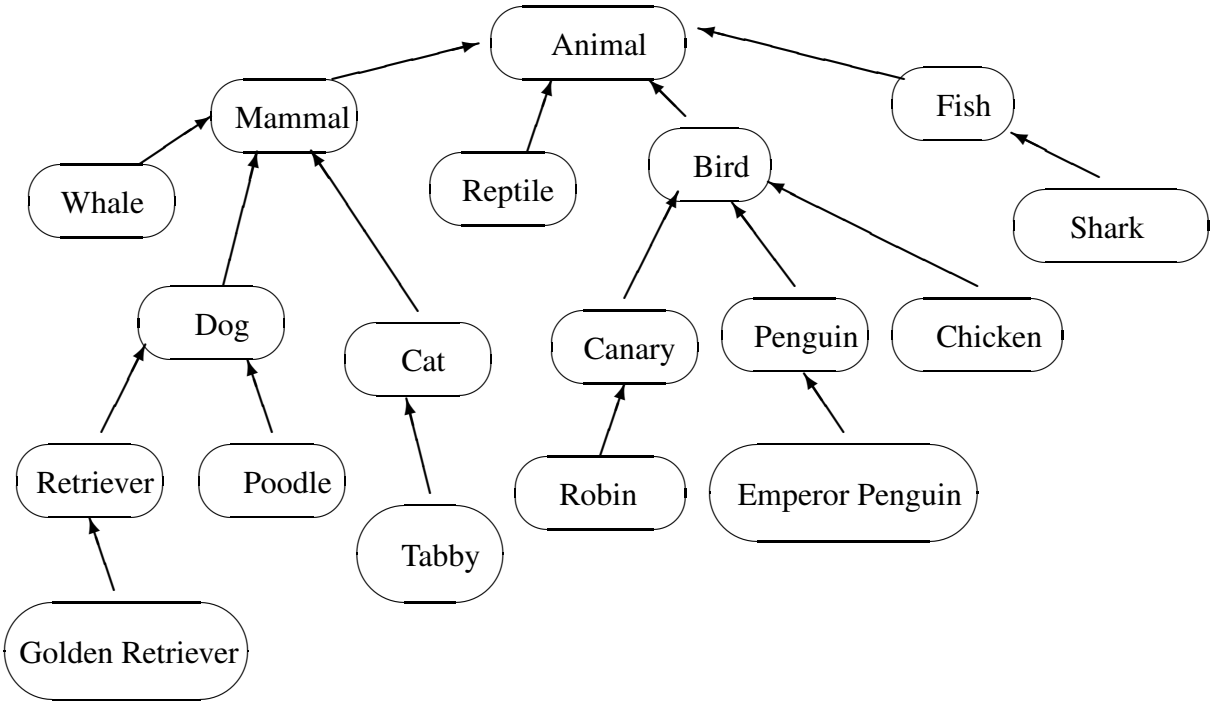
hasAccount(Person,Bank,Amount) – the *Person* has an account at the *Bank* with the balance *Amount*,
lives(Person,City) – the *Person* lives in the *City*,
created(Person,Bank,Month,Year) – the *Person* opened an account at the *Bank* in *Month* of the *Year*.

For simplicity, assume that a person can have only one account at a bank at a time, but can have several accounts at distinct banks. You can enumerate months from 1 to 12. Now pose the following queries to Prolog, and obtain Prolog's answers. You are not allowed to write any rules in this part of your assignment. You can only write atomic statements to implement your KB, and Prolog queries to retrieve answers from your KB. As for syntax of your queries, you can use only negation "not', conjunction, and predicates mentioned above with variables or constants as arguments. You cannot use any other predicates.

1. When did Ann open her account at *Metro Credit Union*?
2. Does anyone have both an account at *Bank of Montreal* and an account at *National Bank of Canada*?
3. Is there a person who has an account at Toronto Dominion and who does not live in Toronto?
4. Who opened an account in August 2019 at the *Royal Bank of Canada*? Have Prolog list all persons who did this by using the ; command (or by clicking the button "more" if you run `tkeclipse` with the GUI).
5. Has anyone opened more than one account with balances less than 1500 in June 2019? Note that $X < Y$ (and $X > Y$) is Prolog's `less than` (`greater than`, respectively) predicate: $X < Y$ succeeds if the number X is less than the number Y. Similarly, $X =< Y$ succeeds if the number X is less than or equal to the number Y.
6. Does any bank have more than two accounts with balances greater than 1000 opened by people who live in New York?
7. Has anyone living outside of Toronto opened accounts at more than one banks in the same year?
8. Who has opened accounts with the same amount at distinct banks in distinct years? Have Prolog list all such persons, banks, and the years when the accounts were opened using the ; command (or by clicking the "more" button on GUI).
9. Find a person from Toronto who has the oldest account at CIBC (note that an account opened in May 2019 is older than an account opened in August 2019 and is older than an account opened in February 2020).
10. Find two distinct people outside of Toronto who have the largest accounts (i.e., the accounts with the two greatest possibly different amounts).

Handing in solutions: (a) An electronic copy of your program with all atomic statements (the name of this file must be **accounts.pl**); (b) An electronic copy of your session with Prolog that shows clearly the queries you submitted to Prolog, and the answers that were returned (the name of this file must be **accounts.txt**)

2 (25 points). Almost all information that is currently available on the World Wide Web (WWW) is intended exclusively for humans, and there are few programs that can do limited processing of this information. The long-term goal of the Semantic Web project is to transform the current WWW so that the information and services are in a machine-processable form. The Semantic Web may create an environment where software agents (sometimes, these computer programs are called "soft-bots" or services) can autonomously perform sophisticated tasks and help humans find, understand, integrate, and use information. In the course, several new important languages have been developed that can be used for representation and automated reasoning about information posted on Semantic Web. One of them is the *Web Ontology Language* (OWL), that has several fragments of varying



expressive power. This Semantic Web activity fits well with the recent research in biological and medical informatics towards unification and integration of medical and biological knowledge. There are medical terminologies that require on the order of 250,000 concepts, some of them are deeply interconnected. The structure of knowledge about anatomy and physiology is that many concepts connect to many others; and notions to be represented range from psychology to molecular biology. While not all terminologies are simple, in this question we concentrate on classes that have single inheritance only.

This problem will exercise what you have learned about rules in Prolog, by having you implement a small *knowledge base* for a simplified collection of classes. One way of representing knowledge about various kinds of objects is to start with a taxonomy, like the following one for animals.

We can represent a taxonomy diagram like this in Prolog using a predicate `isaLink(x,y)` which is true whenever there is direct link in the taxonomy from x up to y. We say in this case that x is a subclass (i.e., *child*) of y or that y is the superclass (i.e., *parent*) of x. (Note that in this question, a class in a taxonomy can have any number of subclasses, but at most one superclass.)

Begin by constructing a taxonomy in Prolog using this `isaLink(x,y)` predicate. Your taxonomy should include all the animals mentioned above, and perhaps some new ones. Next, write a collection of Prolog rules for the following predicates (and any other ones that you feel would be useful)[1]:

- `leaf(x)` is true if x has no subclasses in the taxonomy.
- `sibling(x,y)` is true when x and y are distinct classes that are children of the same parent. For example, in the taxonomy for animals, `dog` and `cat` are siblings, but `retriever` and `cat` are not.
- `grandParent(x,y)` is true when the class x is a child of a child of the class y;
- `isa(x,y)` is true whenever x is equal to y or is a descendant of y in the taxonomy; in other words, every animal of the x kind must be an animal of the y kind. For example, `isa(goldenRetriever,mammal)` is true, `isa(robin,bird)` is also true, but `isa(shark,reptile)` is false.
- `mostSpecificSubsumer(x,y,z)` holds if z is either equal to or an ancestor of x, and equal to or an ancestor of y, but no child of z is; in other words, z is the most specific kind in the taxonomy that animals of both the x and y kind belong to. In the taxonomy above, `mammal` is the most specific subsumer of `poodle` and `cat`.

Handing in solutions. An electronic copy of: (a) your program (the name of the file must be **taxonomy.pl**) that includes all your Prolog rules; (b) your session with Prolog, showing the queries you submitted and the answers returned (the name of the file must be **taxonomy.txt**). It is up to you to formulate a range of queries that demonstrates that your program is working properly.

[1] In this and in the next question, when you write rules (conditional clauses), you are not allowed to use the following commands: "**;**" (disjunction), "**!**" (cut) and "$->$".

3 (35 points).

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

Table 1: Given 14 records representing loan applications.

Advances in data generation and collection are producing data sets of massive size in commerce and a variety of scientific disciplines. The field of data mining grew out of the limitations of current data analysis techniques in handling the challenges posed by these new types of data sets. One of the common problems in data mining is called classification. Classification, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications. Examples include detecting spam email messages based upon the message header and content, categorizing cells as either malignant or benign upon the results of MRI scans, and classifying loan applications as high risk, medium risk or low risk based upon personal data of applicants. The input data for a classification task is a collection of records. Each record, also known as an instance or example, is characterized by several properties that are used to decide whether to place the record in one class or another. A simple yet widely used classification algorithm is a decision tree classifier. Since going into all details of how a decision tree can be constructed out of a given (training) set of records and given properties will take us too far a field, in this question we consider a decision tree that has been already constructed.

For example, consider the problem of estimating an individual *credit risk* on the basis of such properties as *credit history*, *current debt*, *collateral*, and *income*. The table lists a sample of 14 individuals with known credit risks ("*high*", "*moderate*", "*low*"). The decision tree represents this classification in a sense that this tree can correctly classify all the records in the table. In a decision tree, each internal node represents a test on some property, such as *credit history* or *debt*; each possible value of that property corresponds to a branch of the tree. Leaf nodes represent categories (classes), such as "*low risk*" or "*moderate risk*". Once this decision tree has been constructed, the task of deciding where to place a particular record can be completely automated. An individual of unknown type may be classified by traversing this tree: at each internal node, test the individual's value for that property and take the appropriate branch. This continues until reaching a leaf node representing the category.

In machine learning, it can be convenient to represent a decision tree as a set of rules. Each rule includes all tests along a single branch starting from the root towards a leaf node and makes a conclusion where to locate a given record. These rules have several advantages, but we restrain from going into details.

In this question, you have to start with implementing a few (at least 7) records from the given table as a set of atomic statements. Second, you have to write in Prolog the rules that represent **all** branches in the given decision tree. Third, test the records that you have implemented using your rules and compare the classification results with the corresponding leaf nodes in the decision tree. Use the following predicates in your program.

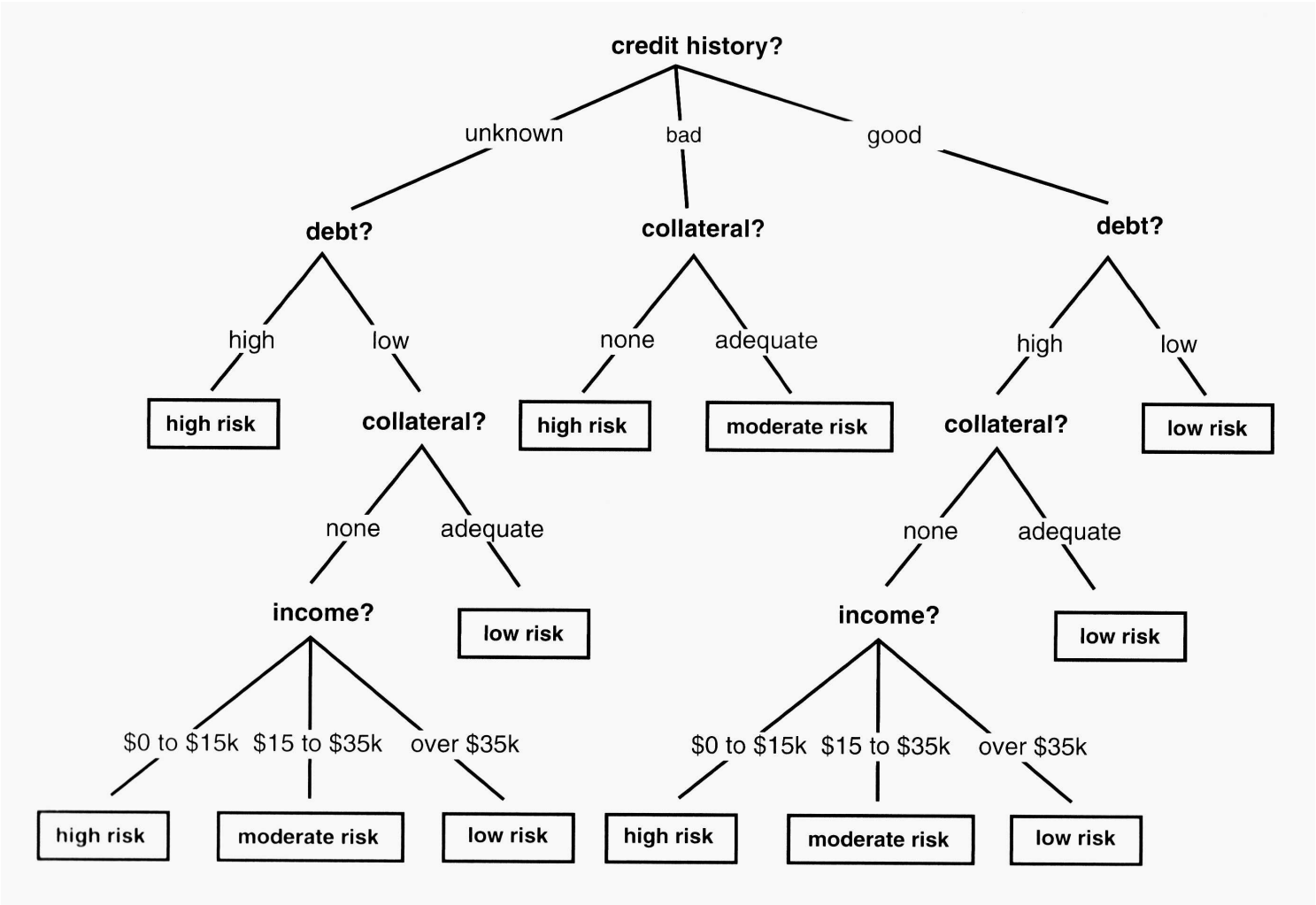


Table 2: The decision tree that correctly assigns classes to all 14 records.

- $risk(Person, Class)$: risk of *Person*'s application is *Class*, where *Class* can be only one out of three: *high, medium, low*.
- $history(Person, Type$: a *Person* has a *Type* credit history, where *Type* can have one of the following values; *bad, unknown, good*.
- $debt(Person, Level)$: a *Person* has a *high* or *low* debt.
- $collateral(Person, Availability)$: whether a *Person* has *none* or *adequate* collateral.
- $income(Person, Group)$: a *Person*'s income is in a *Group*, where group is one of *0_to_15, 15_to_35, over35*.

Handing in solutions. An electronic copy of: (a) your program (the name of the file must be **loans.pl**) that includes all your rules and atomic statements. (b) An electronic copy of your session with Prolog that shows clearly the queries you submitted to Prolog, and the answers that were returned (the name of this file must be **loans.txt**);

How to submit this assignment. Read regularly *Frequently Answered Questions* and replies to them that are linked from the Assignments Web page at

`http://www.scs.ryerson.ca/~mes/courses/cps721/assignments.html`

If you write your code on a Windows machine, make sure you save your files as plain text that one can easily read on Linux machines. Before you submit your Prolog code electronically make sure that your files do not contain any extra binary symbols: it should be possible to load `accounts.pl` or `taxonomy.pl` or `loans.pl` into a recent release 6 of ECLiPSe Prolog, compile your program and ask testing queries. TA will mark your assignment using ECLiPSe Prolog. If you run any other version of Prolog on your home computer, it is your responsibility to make sure that your program will run on ECLiPSe Prolog (release 6 or any more recent release), as required. For example, you can run a command-line version of */opt/ECLiPSe/bin/x86_64_linux/eclipse* executable on moon remotely from your home computer to test your program (read handout about running *ECLiPSe Prolog*). To submit files electronically do the following. First, create a **zip** archive:

```
zip yourLoginName.zip accounts.pl accounts.txt taxonomy.pl taxonomy.txt loans.pl loans.txt
```

where `yourLoginName` is the login name of the person who submits this assignment from a group. Remember to mention at the beginning of each file *student, section numbers* and *names* of all people who participated in discussions (see the course management form). You may be penalized for not doing so.

Second, upload your file (one file only!) **yourLoginName.zip**
(make sure it includes **all** files) to D2L into "Assignment 1" folder. Improperly submitted assignments will **not** be marked. In particular, you are **not** allowed to submit your assignment by email to a TA or to the instructor.

Revisions: If you would like to submit a revised copy of your assignment, then upload your file again. (The same person must upload it.) A new copy of your ZIP file will override the old copy. You can upload new versions as many times as you like and you do not need to inform anyone about this. Don't ask your team members to upload your assignment, because the TA will be confused which version to mark: only one person from a group should submit different revisions of the assignment. The time stamp of the last file you submit will determine whether you have submitted your assignment in time.