CPS721: Assignment 1

Due: September 26, 2023, 9pm
Total Marks: 100 (worth 4% of course mark)
You MUST work in groups of 2 or 3

Late Policy: The penalty for submitting even one minute late is 10%. Assignments are not accepted more than 24 hours late.

Clarifications and Questions: Please use the discussion forum on the D2L site to ask questions as they come up. These will be monitored regularly. Clarifications will be made there as needed. A Frequently Asked Questions Page will also be created. You may also email your questions to your instructor, but please check the D2L forum and frequently asked questions first.

Collaboration Policy: You can only discuss this assignment with your group partners or with your CPS721 instructor. By submitting this assignment, you acknowledge that you have read and understood the course policy on collaboration as stated in the CPS721 course management form.

PROLOG Instructions: When you write your rules in PROLOG, you are not allowed to use ";" (disjunction), "!" (cut), and "->" (if-then). You are only allowed to use ";" to get additional responses when interacting with PROLOG from the command line. Note that this is equivalent to using the "More" button in the ECLiPSe GUI.

We will be using ECLiPSE Prolog release 6 to mark the assignments. If you run any other version of PROLOG, it is your responsibility to check that it also runs in ECLiPSE Prolog release 6.

Submission Instructions: You should submit ONE zip file called `assignment1.zip`. I should contain 6 files:

- `q1_albums.pl`, `q1_albums_queries.txt`
- `q2_grocery_bill.pl`, `q2_queries.txt`
- `q3_intersection.pl`, `q3_queries.txt`

All these files have been given to you and you should fill them out using the format described. Your submission should not include any other files. If you submit a `.rar`, `.tar`, `.7zip`, or other compression format aside from `.zip`, you will lose marks. All submissions should be made on D2L. Submissions by email will not be accepted.

As long as you submit your assignment under the name `assignment1.zip` your group will be able to submit multiple times as it will overwrite an earlier submission. You do not have to inform anyone if you do. The time stamp of the last submission will be used to determine the submission time. Do not submit multiple `zip` files with different names. If you do, we will use the last submitted one, but you may lose marks.

If you write your code on a Windows machine, make sure the files are saved on plain text and are readable on Linux machines. Ensure your PROLOG code does not contain any extra binary symbols and that they can be compiled by ECLiPSE Prolog release 6.

1 A PROLOG Knowledge Base [30 marks]

For this problem, you will be making a knowledge base about songs and then perform queries on it. The knowledge base and rules will be stored in the given file `q1_albums.pl`. Your test queries should be added to `q1_albums_queries.txt` in the sections indicated. Finally, a log of your interaction when applying those queries to the knowledge base should be put at the end of the `q1_albums_queries.txt` file.

a. [10 marks] Create a knowledge base by adding atomic propositions to the file `q1_albums.pl`. The atomic propositions should ONLY use the following predicates:

- `albumArtist(AlbumName, ArtistName)` - an artist of the album.
- `albumLabel(AlbumName, LabelName)` - the record label (*ie.* company) that released the album.
- `albumYear(AlbumName,Year)` - the year the album was released.

You should add 10-15 sentences per predicate about the albums. The data does not need to be "real", in the sense that you can make up album names, artist names, years, and record labels. The purpose of the knowledge base is to get you practice building a knowledge base. You will also be using it for testing your queries in part b. Thus, you should add facts that allow you to test your queries effectively.

A few additional notes:

- A predicate called `year(Y)` can also be used, which lists valid years. We have provided atomic statements which use this statement covering the years from 2000 to 2023. For the purposes of this assignment, stick to those years only and do not edit the given list.
- Remember to use lower case for artist names, album names, and record label names. Do not put these in quotes. Words should be separated by the underscores. For example, to refer to the band "Neutral Milk Hotel", use the constant `neutral_milk_hotel`.
- For the purposes of this assignment, ignore accents in names or punctuation in album titles. For example, Ariana Grande's album "Thank U, Next" should appear as `thank_u_next`. Similarly, avoid any album or artist names that use non-alphabetic or numeric symbols (like Ed Sheeran's album ÷).

All atomic propositions should be added to the file `question1_albums.pl` in the correct sections. You will lose marks if you do not put propositions in the correct place.

b. [20 marks] Create queries for each of the 10 statements below and add them to the file `q1_albums_queries.pl`. Ensure that you put each query under the correct comment as otherwise you may lose marks. You can only use the predicates listed above with variables or constants as arguments, conjunction, and "not" (*ie.* negation) in your queries. You may also use $<, >, =<$, or $>=$, which PROLOG uses for less than, more than, etc. Keep in mind that when using a predicate $X < Y$, both X and Y must be instantiated before the comparison.

You should test all your queries in an interaction with your knowledge base and then submit the text at the end of the `q1_albums_queries.pl` file.

1. (1 mark) Did the record label Republic release an album called "Midnights"?
2. (1 mark) Who was the artist that released the album "Renaissance"?
3. (2 marks) Was there an artist who released albums in both 2010 and 2016, and if so, who was it?

4. (2 marks) Was there an album that was released in the year 2012 that was not released by either the record label Columbia or the record label Republic? In your interaction file, list all such albums using the ";" command if you are using the command line or the "More" button if you are using the GUI.

5. (2 marks) Did any artist release two different albums, each with a different record label?

6. (2 marks) Did any artist release more than one album with the same record label?

7. (2 marks) Did the artist who released an album called "30" release any albums in years before that one? In your interaction file, list all such albums using the ";" command if you are using the command line or the "More" button if you are using the GUI.

8. (2 marks) Was there a year from 2000 to 2023 (inclusive) that the record label Republic did not release an album?

9. (2 marks) Was there an artist that released an album in 3 consecutive years starting in 2018 or later?

10. (2 marks) Did any record label release exactly one album in 2015?

11. (2 marks) What was the last album released by Drake?

Note that your queries should capture the logic of the given statement, not just return the correct statement for your knowledge base. For example, if we changed the knowledge base, your queries should still return the correct answers for the new knowledge base. Do not worry if your queries return duplicate answers if you request multiple answers. However, you will lose marks if the query ever returns incorrect answers for your KB or a different KB used for testing.

2 Arithmetic in PROLOG [30 marks]

This questions involves numerical calculations in PROLOG. The knowledge base and rules will be stored in the given file `q2_estore.pl`. Your test queries should be stored in `q2_estore_queries.txt`. Finally, a log of your interaction when applying those queries to the knowledge base should be included at the end of the `q2_estore_queries.txt` file. Both of these files need to be included as part of the submission.

a. [**10 marks**] You will be calculating the bill at an e-store on three products: `laptop`, `monitor`, and `keyboard`. For this step, you should create a knowledge base that contains atomic statements that use the following predicates:

- `cost(Product, Cost)` - defines the cost of the product.
- `numPurchased(Product, Count)` - indicates the number of that product that has been purchased.
- `shippingCost(Product, Cost)` - indicates the `Cost` to ship each unit of a given `Product` for regular 7-day shipping.
- `taxRate(Rate)` - the tax rate.
- `freeRegularShippingMin(Amount)` - the minimum amount of pre-tax purchases a user must make in order to get free 7 day shipping.
- `freeExpressShippingMin(Amount)` - the minimum amount of pre-tax purchases a user must make in order to get free express shipping.

All products should have a positive cost and non-negative values for the number purchased. You may set the tax rate to any value greater than 0 but less than 1 (*ie.* the tax rate of 13% would be 0.13). The values for free shipping should be non-negative, but may be anything you choose.

All atomic propositions should be added to the file `question2_estore.pl` in the correct sections. You will lose marks if you do not put propositions in the correct place.

b. [**15 marks**] Add rules to `q2_estore.pl` that accomplish the following:

- Add a rule `subtotal(Sub)` which calculates the sum of the costs (`Sub`) of all three products being purchased, not including tax and shipping. This will depend on the cost of each product and the number of each that is purchased.
- Add a rule `calculateBaseShipping(ShippingCost)` which calculates the cost of over all products of regular 7-day shipping, but does not account for if the customer has earned free shipping.This rule will be used as a helper below, but should only depend on the shipping cost of each product and the number of each purchased. Note that `ShippingCost` should only include the cost of shipping, not of the products themselves.
- Add rule `calculateShipping(ShippingType, ShippingCost)` which takes in a `ShippingType` which is either `regular` or `express` and calculates the cost (`ShippingCost`) for shipping the current order, depending on the shipping type and whether or not the customer has earned free shipping. The cost of express shipping a product is 1.5 times the cost of shipping it for regular 7-day shipping. You can use your `subtotal` and `calculateBaseShipping` predicates. Note that `ShippingCost` should only include the cost of shipping, not of the products themselves.
- Add a rule `totalCost(ShippingType, Cost)` that calculates the total cost for all the products including shipping and tax. Note that tax applies to the shipping cost as well.

All rules should be added to the file `question2_estore.pl` in the correct sections. You may lose marks if you do not put them in the correct place. We will be testing these rules using a variety of knowledge bases, so make sure they capture the logic of the rule, and do not just work correctly on your particular knowledge base.

c. [**5 marks**] Create at least 6 queries that test the `subtotal`, `calculateBaseShipping`, `calculateShipping`, and `totalCost` rules. You should not use the ";" command in these queries. Any queries of your choosing on these rules is allowed, as long as each is tested at least once. You should also add a comment above each query stating what the query checks in plain English.

The queries should be written in the file `q2_estore_queries.txt` . At the end of this file, submit your interaction with the knowledge base as well.

3 Recursive Rules [40 marks]

Paul Erdős was a mathematician who was famous for how prolific he was, how much his work crossed different areas of mathematics, and how many other mathematicians he worked with. To this day, mathematicians and computer scientists will refer to their *Erdős* number which calculates how many collaborations a person is away from Paul Erős. This number is calculated as follows[1]:

- Paul Erős has an Erdős number of 0.
- Anyone who wrote an publication with Paul Erős has an Erdős number of 1.
- A person's Erdős number is then given by 1 plus the minimum Erdős number of anyone they wrote an article/publication with.

This concept can be generalized to any pair of scientists/mathematicians such that we can try to find the length of the shortest path of collaborators from one to the other. Below, you will solve a simplified task: given a pair of scientists/mathematicians and a number `MaxDist`, your rule will answer the question of whether or not there is a collaboration path of length no more than `MaxDist` between the two.

This question involves writing recursive rules for this problem. You will write a knowledge base and rules which are to be stored in the given file `q3_collab.pl`. You are also to write test queries and include the results of your interaction in `q3_collab_queries.txt`. Use the format specified in the file or you will lose marks.

a. [10 marks] Create a knowledge base containing at least 5 publications and 10 authors. Both the publications and authors may be fictional. The knowledge base should contain atomic statements for the following predicates:

- `articleAuthor(Article, Author)` states that `Author` is one of the authors of the article with name `Article`. An article may have more than one author. As in Question 1, ignore accents and punctuation and do not put author or article names in quotes (ie. `paul_erdos` was an author of `on_sets_of_distances_of_n_points`).
- `articleTopic(Article, Topic)` states that `Topic` is a topic of the given `Article`. For example, the topic may be `ai` or `cybersecurity` or `programming`. An article may have more than one topic.

As in the previous questions, this knowledge base will be used for testing your rules in the following parts. All atomic propositions should be added to the file `q3_collab.pl` in the correct sections. You may lose marks if you do not put propositions in the correct place.

b. [10 marks] Write the rules for the following predicate and add them to the file `q3_collab.pl` in the correct sections:

```
collabDist(Author1, Author2, MaxDist)
```

This query will succeed if and only if there is a path of length at most `MaxDist` collaboration steps between the two given authors. Notice that your rule does *not* have to compute the minimum distance between the two, just whether there is a path no bigger than the given value.

For example, suppose we have a knowledge base with 3 authors (`tom`, `jennifer`, and `tina`) and two articles (`article` authored by `tom` and `jennifer`, and `article2` authored by `jennifer` and `tina`). Then `collabDist(tom, tina, 2)` should succeed, as should `collabDist(tom, tina, 3)`, `collabDist(tom, tina, 4)`, etc.. However, `collabDist(tom, tina, 1)` would not succeed.

[1]The "Bacon number" is a similar concept for movies. That number measures how many "movies" an actor is away from being in a movie with Kevin Bacon. If a mathematician has appeared in a movie, they may even have an "Erdős-Bacon" number, which is the sum of your Erdős and Bacon number (assuming they are both finite)."

Additional notes for part b:

- In all queries, `MaxDist` will be a non-negative integer and *not* a variable. Thus, your rules should be able to handle queries of the form `collabDist(tom, jennifer, 2)` or `collabDist(X, tina, 5)`, but we will *not* test it with queries such as `collabDist(tom, jennifer, X)`.
- Your rules should also succeed if `Author1 = Author2` for any `MaxDist` ≥ 0, provided that that author has written at least one article. For example, `collabDist(tom, tom, 2)` should succeed in the above example, but `collabDist(ron, ron, 2)` should fail.
- If multiple answers are requested using ";" or "More" it is ok if your rules return duplicate answers. However, they should never return incorrect answers.
- For the purposes of this assignment, a collaboration path may include the same collaborator multiple times. For example, `tom` to `jennifer` to `tina` to `jennifer` is a valid path.
- HINT: Due to `MaxDist` variable, you should not have to worry about infinite paths of the form `tom`, `jennifer`, `tom`, `jennifer`. ... Think about why that is, but you do not need to submit writeup of your reasoning.

c. [**15 marks**] You will now write the following new predicate

```
collabDistWithAI(Author1, Author2, MaxDist, AI)
```

Just as before, your predicate should determine if there is a collaboration path between `Author1` and `Author2` that has at most `MaxDist` steps. The variable `AI` will then take on value of either `none` or `at_least_one`. If `AI` is `none`, then the query will only succeed if there is a collaboration path from `Author1` to `Author2` such that *none* of the papers along the path have the topic of `ai`. If `AI` takes on the value `at_least_one`, then *at least one* of the collaborations along the collaboration path will involve a paper whose topic is `ai`.

For example, suppose `tom` wrote `article1` with `jennifer`, and `jennifer` wrote `article2` with `tina`. Then `collabDistWithAI(tom, tina, 2, at_least_one)` would succeed if at least one of `article1` or `article2` (but not necessarily both) have a topic of `ai`. The query `collabDistWithAI(tom, tina, 2, none)` would succeed if neither `article1` or `article2` have a topic of `ai`.

Additional notes for part c:

- The same notes given for `collabDist` also apply to `collabDistWithAI`.
- If there are multiple collaboration paths between the authors, then `collabDistWithAI` may succeed for both `none` and `at_least_one`. Consider the example above. If both `article1` and `article2` are not AI articles, then `collabDistWithAI(tom, tina, 2, none)` will succeed, regardless of what other articles there are. If additionally, `article3` is on the topic of AI and was written by both `tom` and `tina`, then `collabDistWithAI(tom, tina, 2, at_least_one)` will also succeed.
- A query like `collabDistWithAI(tom, tom, 0, none)` should only succeed if `tom` has at least one non-AI article. Similarly, `collabDistWithAI(tom, tom, 0, at_least_one)` should only succeed if `tom` has at least one AI article.
- If `tom` has only non-AI articles, it may still be possible for a query such as `colalbDistWithAI(tom, tom, 3, at_least_one)` to succeed, but this will depend on what articles `tom`'s collaborators have.
- Be careful about how `collabDistWithAI(tom, tom, 0, at_least_one)` interacts with other rules. For example, if `article1` is not an AI article and it is the only article that `tom` wrote with `jennifer`, then `collabDistWithAI(tom, jennifer, 1, at_least_one)` should fail even if `jennifer` has also written an AI article.

- HINT: Get the new predicate rule working for the `none` case first, and then consider how you need to change it to handle `at_least_one`. You will get part marks if you only get the `none` case working.

c. [5 marks] Create at least 10 queries that test `collabDist` and `collabDistWithA`. You should not use the ";" command in these queries, though you may use it to get multiple answers. Any queries of your choosing on these rules is allowed, as long as each is tested at least once. You should also add a comment above each query stating what the query checks in plain English.

The queries should be written in the file `q3_queries.txt` . At the end of this file, submit your interaction with the knowledge base as well.