Middle East Technical University   Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring 2023-2024

Programming Exam 6

Due date: 4 June 2024, Tuesday, 23.59

1 Problem Definition

Your millennial friend is yapping about the old web again, how good it was, how everybody had their own blogs, and all that. He is currently using an RSS feed to follow his remaining favourite bloggers. You, being a child of the internet with algorithmic feeds, decide to implement some extended functionality to your friend's preferred way of reading blogs. To achieve this, you will implement some Prolog methods that focus on blogger and post recommendations.

1.1 Knowledge Base Predicates

All knowledge will be provided in the knowledge base with the following predicates:

```
blogger(BloggerNick).
```

The `blogger` predicate is used to define bloggers with available RSS feeds to follow, where `BloggerNick` is the nickname of the blogger.

```
reader(ReaderNick, ListOfInterestTopics).
```

The `reader` predicate is used to define readers that use the RSS feed. Its format is given above, where `ReaderNick` is the nickname of the reader, and `ListOfInterestTopics` is a list of topics the reader is interested in.

```
posted(BloggerNick, PostID).
```

The `posted` predicate is used to capture the which posts are published by which bloggers. Its format is given above, where `BloggerNick` is the nickname of the blogger, and `PostID` is a unique ID given for each post.

```
follows(ReaderNick, BloggerNick).
```

The `follows` predicate is used to keep track which users follow which bloggers. Its format is given above, where `ReaderNick` is the nickname of the reader, and `BloggerNick` is the nickname of the blogger.

```
blogpost(PostID, ShortPostName, Topic, Date).
```

The `blogpost` predicate is used to define details of the posts. Its format is given above, where `PostID` is a unique ID given for a post, `ShortPostName` is a unique nickname given for a post, `Topic` is a singular category that the blogpost is about, and `Date` is a number that represent the publishing date of the post in YYYYMMDD format.

```
alreadyread(ReaderNick, PostID).
```

The `alreadyread` predicate is used to capture the which posts are already read by which users. Its format is given above, where `ReaderNick` is the nickname of the reader, and `PostID` is a unique ID given for each post.
The example knowledge base given to you in kb.pl file can be seen below:

```
blogger(crimeow).
blogger(internetbaby).
blogger(tashrants).
blogger(basicape).
blogger(beyazfutbolfan).

reader(tijocat, [cyberCrimes, popCulture, fashion, science, food]).
reader(justaguy, [cryptocurrency, finance, sports]).

posted(crimeow, pid001).
posted(crimeow, pid002).
posted(internetbaby, pid021).
posted(internetbaby, pid022).
posted(basicape, pid031).
posted(beyazfutbolfan, pid032).
posted(tashrants, pid041).
posted(tashrants, pid042).

follows(tijocat,crimeow).
follows(tijocat,internetbaby).
follows(justaguy, basicape).

blogpost(pid001, spywaredump, cyberCrimes, 20230922).
blogpost(pid002, spywaredumpagain, cyberCrimes, 20231028).
blogpost(pid021, metgala2024flops, fashion, 20240510).
blogpost(pid022, celebdivorcewatch, celebGossip, 20240116).
blogpost(pid031, getricheasynow, cryptocurrency, 20240313).
blogpost(pid032, amanhocamdur, sports, 20210317).
blogpost(pid041, culturewrap2023, popCulture, 20231231).
blogpost(pid042, futureofacademia, science, 20221231).

alreadyread(tijocat, pid001).
alreadyread(tijocat, pid021).
alreadyread(justaguy, pid031).
```

2 Specifications

In this exam, you are expected to implement different predicates with varying difficulties. All predicates will be queried with different arguments given as variables, but not all arguments will be queried for all predicates. Querying details for each predicate can be seen in the example queries for that predicate.

Expected implementations of some predicates aim to familiarize you with some common built-in predicates of Prolog. You do not necessarily need to use them, but using them is advised. The built-in predicates that might come in handy for this part are as follows:

- `findall/3` https://www.swi-prolog.org/pldoc/man?predicate=findall/3
- `length/2` https://www.swi-prolog.org/pldoc/man?predicate=length/2

You are free to use other built-ins, or you can implement everything by yourself. You can navigate around the shared links to see the documentation for other built-ins.

2.1 `is_blogpost_written_by/2` - 5 points

Two argument predicate where `BloggerNick` is the nickname of the blogger, `ShortPostName` is the nickname of the post. A blogpost is written by the given blogger if there is a `posted` knowledge that matches the `PostID` of the post with the given `ShortPostName` between the given blogger. Predicate has the form:

```
is_blogpost_written_by(BloggerNick,ShortPostName).
```

Example queries:

```
?- is_blogpost_written_by(crimeow, spywaredump).
true.

?- is_blogpost_written_by(crimeow, getricheasynow).
false.
```

- This predicate **will not be queried with its arguments as variables**, so you do not need to worry about the behavior of such queries.

2.2 `has_post_in_topic/2` - 5 points

Two argument predicate where `BloggerNick` is the nickname of the blogger, `Topic` is the name of the topic. A blogger has a post with the given topic if there is at least one post belonging to the blogger of the given topic. Predicate has the form:

```
has_post_in_topic(BloggerNick,Topic).
```

Example queries:

```
?- has_post_in_topic(crimeow,cyberCrimes).
true.

?- has_post_in_topic(crimeow,fashion).
false.
```

- This predicate **will not be queried with its arguments as variables**, so you do not need to worry about the behavior of such queries.
- If the blogger has more than one post with the given topic, you should return true once. (Hint: you may need to use ”!”)

2.3 `get_follower_count_of_blogger/2` - 10 points

Two argument predicate where `BloggerNick` is the nickname of the blogger, `FollowerCount` is the number of followers the blogger has. Followers are defined by the `follows(ReaderNick, BloggerNick)` knowledge that matches the `BloggerNick` of the given blogger. Predicate has the form:

```
get_follower_count_of_blogger(BloggerNick,FollowerCount).
```

Example queries:

```
?- get_follower_count_of_blogger(crimeow,FollowerCount).
FollowerCount = 1.

?- get_follower_count_of_blogger(tashrants,FollowerCount).
FollowerCount = 0.
```

- This predicate **will only be queried with `FollowerCount` argument as a variable**, so you do not need to worry about the behavior of other queries.

2.4 `get_follower_count_of_blogger/2` - 10 points

Two argument predicate where `Topic` is the name of the topic, `PostCount` is the number of posts that is written in the given topic. Predicate has the form:

```
get_post_count_of_topic(Topic,PostCount).
```

Example queries:

```
?- get_post_count_of_topic(cyberCrimes,PostCount).
PostCount = 2.

?- get_post_count_of_topic(food,PostCount).
PostCount = 0.
```

- This predicate **will only be queried with `PostCount` argument as a variable**, so you do not need to worry about the behavior of other queries.

2.5 `filter_posts_by_date/4` - 20 points

Four argument predicate where `ListOfPostIDs` is the list of posts to be filtered, `OldestDate` and `NewestDate` are the oldest and newest dates accepted for a post to be published respectively, and `ListOfFilteredPostIDs` are the filtered post that are published between and on the given dates. Predicate has the form:

```
filter_post_by_date(ListOfPostIDs,OldestDate,NewestDate,ListOfFilteredPostIDs).
```

Example queries:

```
?- filter_posts_by_date([pid041,pid042],20230101,20231231,ListOfFilteredPostIDs).
ListOfFilteredPostIDs = [pid041] ;
false.

?- filter_posts_by_date([pid001,pid002,pid021,pid022,pid031,pid032,pid041,pid042↩
   ],20140101,20170101,ListOfFilteredPostIDs).
ListOfFilteredPostIDs = [] ;
false.
```

- This predicate **will only be queried with `ListOfFilteredPostIDs` argument as a variable**, so you do not need to worry about the behavior of other queries.

2.6 `recommend_posts/2` - 25 points

Two argument predicate where `ReaderNick` is the name of the reader to recommend posts to, and `ListOfRecommendedPosts` are the list of PostIDs of the recommended posts. You need to find posts that are written in a topic that the reader is interested in, but has not read yet. Predicate has the form:

```
recommend_posts(ReaderNick,ListOfRecommendedPosts).
```

Example queries:

```
?- recommend_posts(tijocat,ListOfRecommendedPosts).
ListOfRecommendedPosts = [pid002, pid041, pid042] ;
false.

?- recommend_posts(justaguy,ListOfRecommendedPosts).
ListOfRecommendedPosts = [pid032] ;
false.
```

- This predicate **will only be queried with `ListOfRecommendedPosts` argument as a variable**, so you do not need to worry about the behavior of other queries.

2.7 `recommend_bloggers/2` - 25 points

Two argument predicate where `ReaderNick` is the name of the reader to recommend bloggers to, and `ListOfRecommendedBloggers` are the list of nicknames of the recommended bloggers. You need to find bloggers that has written at least one post in a topic that the reader is interested in, but the reader does not follow. Predicate has the form:

```
recommend_bloggers(ReaderNick,ListOfRecommendedBloggers).
```

Example queries:

```
?- recommend_bloggers(tijocat,ListOfRecommendedBloggers).
ListOfRecommendedBloggers = [tashrants] ;
false.

?- recommend_bloggers(justaguy,ListOfRecommendedBloggers).
ListOfRecommendedBloggers = [beyazfutbolfan] ;
false.
```

- This predicate **will only be queried with `ListOfRecommendedBloggers` argument as a variable**, so you do not need to worry about the behavior of other queries.

3 Regulations

- **Implementation and Submission:** The template files are available in the Virtual Programming Lab (VPL) activity called "PE6" on ODTUCLASS. At this point, you have two options:
  - You can download the template files, complete the implementation, and test it with the given sample I/O on your local machine. Then submit the same file through this activity.
  - You can directly use the editor of the VPL environment by using the auto-evaluation feature of this activity interactively. Saving the code is equivalent to submitting a file.

  Please make sure that your code runs on ODTUCLASS. There is no limitation in running your code online. The last save/submission will determine your final grade.

- **Programming Language:** You must code your program in Prolog. Your submission will be run with swipl on ODTUCLASS. You are expected to make sure your code runs successfully with swipl on ODTUCLASS.

- **Modules:** You are not allowed to import any modules. However, you are allowed to use the built-in predicates present in Prolog or define your own predicates.

- **Cheating: This assignment is designed to be worked on individually.** You are free to utilize recitation materials and shared old homeworks, but use of any LLMs (chatgpt, copilot, the other one that you are thinking about...) for implementations are strictly forbidden.

- **Evaluation:** Your program will be evaluated automatically using "black-box" testing, so make sure to obey the specifications. No erroneous input will be used. Therefore, you don't have to worry about invalid cases.

  Important Note: The given sample I/O's are only to ease your debugging process and NOT official. Furthermore, it is not guaranteed that they cover all the cases of required functions. As a programmer, it is your responsibility to consider such extreme cases for the functions. Your implementations will be evaluated by the official test cases to determine your final grade after the deadline.