Middle East Technical University Department of Computer Engineering

# CENG 242

Programming Language Concepts

Spring 2020-2021

Programming Exam 8

Due date: June 17, 2021, Thursday, 23:59

## 1 Problem Definition

In this lab exam, you are going to implement predicates that operate on a knowledge base with people. These people have their names, ages and hobbies.

### 1.1 Knowledge Base Predicate

The **person** predicate is used to define people and their attributes. Its format is given below:

```
person(Name, Age, Hobby).
```

The example knowledge base given to you in kb.pl file can be seen below.

```
person(joseph, 27, reading).
person(jessica, 32, crafting).
person(michael, 22, reading).
person(william, 33, reading).
person(elizabeth, 30, television).
person(jennifer, 38, crafting).
person(patricia, 33, bird_watching).
person(charles, 39, bird_watching).
person(david, 31, bird_watching).
person(mary, 25, crafting).
person(barbara, 25, reading).
person(richard, 32, travelling).
person(james, 22, fishing).
person(susan, 32, reading).
person(karen, 40, bird_watching).
person(sarah, 25, crafting).
person(linda, 21, reading).
person(john, 28, reading).
person(thomas, 23, bird_watching).
person(robert, 22, television).
```

## 2 Specifications

In this lab exam, you are expected to code 3 different predicates with varying difficulties.

## 2.1 The sum_age predicate - 30 pts

It has two arguments. The first one is a list containing names. The second one is the sum of all ages that belong to the people in that list.

Format of the predicate is given below:

```
sum_age(NameList, TotalAge).
```

Examples:

```
?- sum_age([], TotalAge).
TotalAge = 0.

?- sum_age([joseph, jessica, michael], TotalAge).
TotalAge = 81.

?- sum_age([susan, james], TotalAge).
TotalAge = 54.
```

## 2.2 The max_age_of_hobby predicate - 35 pts

It has 3 arguments. The first one is a list containing names. The second one is the specified hobby. The third one is the maximum age of the people in the first list that has the specified hobby. If there is no such person in the list, the result will be 0. (You may want to use \= operator to test for not-matching of two terms. For example, X\=Y evaluates to true if and only if X=Y evaluates to false.)

Format of the predicate is given below:

```
max_age_of_hobby(NameList, Hobby, MaxAge).
```

Examples:

```
?- max_age_of_hobby([], reading, MaxAge).
MaxAge = 0.

?- max_age_of_hobby([jessica], reading, MaxAge).
MaxAge = 0.

?- max_age_of_hobby([joseph], reading, MaxAge).
MaxAge = 27 ;
false.

?- max_age_of_hobby([joseph, jessica], reading, MaxAge).
MaxAge = 27 ;
false.

?- max_age_of_hobby([charles, joseph, jessica, william], reading, MaxAge).
MaxAge = 33 ;
false.

?- max_age_of_hobby([barbara, elizabeth, robert, thomas, karen], television, ↩
    MaxAge).
MaxAge = 30 ;
false.
```

### 2.3 The person_in_range predicate - 35 pts

It has 4 arguments. The first one is a list containing names. The second one is the minimum age threshold and the third one is the maximum age threshold (inclusive). The fourth argument is the result list which is a list of names from the NameList (the first argument) that resides in the specified age range. The order of the names in the result list should be the same as the given name list.

Format of the predicate is given below:

```
person_in_range(NameList, Min, Max, ResultList).
```

Examples:

```
?- person_in_range([], 20, 25, Res).
Res = [].

?- person_in_range([michael], 20, 25, Res).
Res = [michael] ;
false.

?- person_in_range([joseph, michael], 20, 25, Res).
Res = [michael] ;
false.

?- person_in_range([joseph, michael, mary], 20, 25, Res).
Res = [michael, mary] ;
false.

?- person_in_range([michael, william, james], 22, 22, Res).
Res = [michael, james] ;
false.
```

## 3 Regulations

- **Implementation and Submission**: The template files are available in the Virtual Programming Lab (VPL) activity called "PE8" 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 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 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 base predicates present in prolog or define your own predicates.

- **Cheating**: Using code from any source other than your own is considered cheating. This includes but not limited to internet sources, previous homeworks/exams, and friends. In case of cheating, the university regulations will be applied.

- **Grading**: You can evaluate your code interactively on Cengclass. However, note that the evaluation test cases are only provided to assist you and are different from the test cases that will be used during grading. Thus, the grade resulting from the evaluation may not be your final grade.
- **Late Submission**: There is no late submission.