

CmpE 320 – Spring 2008 – Programming Project #1



This project will test your knowledge of functional programming and Scheme. Your code will be graded on correctness, readability, testing strategy, efficiency, documentation and the use of functional programming style. Your task is to write Scheme functions (statements) for extracting information about the relationship of theoretical computer science PhD graduates taken from The TCS Genealogy project on <http://sigact.acm.org/genealogy/>.

Problem Description

You will extract different sorts of information from a database of people who are the professors of theoretical computer science. We encoded the database in Scheme as a list of entries, with one entry for each person. Each entry will be a list in the following form:

```
(<person> <university> <year> <student1> ... <studentN>)
```

which indicates that <person> is a PhD graduate from <university> in <year> and after being a professor, s/he supervised <student₁> ... <student_N>.

A portion of the sample database is as follows:

```
(define SCIENTISTS
 '( (norman_i_badler university_of_toronto 1975 joseph_o_rourke bulent_ozguc)
   (joseph_o_rourke university_of_pennsylvania 1980 alok_aggarwal subhash_suri)
   (alok_aggarwal johns_hopkins_university 1984 dina_kravets)
   (dina_kravets massachusetts_institute_of_technology 1992)
   (subhash_suri johns_hopkins_university 1987)
   (bulent_ozguc university_of_pennsylvania 1979 veysi_isler)
   (veysi_isler bilkent_university 1995)
   (gary_miller university_of_california_at_berkeley 1975 steve_quatterry susan_landau tom_leighton)
   (steve_quatterry carnegie_mellon_university 1995)
   (susan_landau massachusetts_institute_of_technology 1983)
   (tom_leighton massachusetts_institute_of_technology 1981 thang_bui dina_kravets)
   (thang_bui massachusetts_institute_of_technology 1986)
   (pafnuty_lvovich_chebyshev st_petersburg_university 1849 andrei_andreevich_markov)
   (andrei_andreevich_markov st_petersburg_university 1884 georgy_fedoseevich_voronoi)
   (georgy_fedoseevich_voronoi st_petersburg_university 1897)
 )
 )
```

For example, the first entry can be read as “*Norman I. Badler* took his PhD from *University of Toronto* in 1975 and *Joseph O’Rourke* and *Bülent Özgüç* are supervised by him during their PhD theses.”.

Observe that:

- The <person>, <university> and <year> parts of an entry are required in all entries but the remaining part (<student₁> ... <student_N>.) varies from no students to N students. For example *Voronoi* seems to be a supervisor of nobody in the given database.
- If a person is in the student list of a professor, then s/he is also a professor in an entry. For example, *Tom Leighton* supervised *Thang Bui* and *Dina Kravets*, so there must be entries starting with *Thang Bui* and *Dina Kravets* as there are in the given database.
- A PhD student may have more than one supervisor. For example *Dina Kravets’* PhD thesis is supervised by *Alok Aggarwal* and *Tom Leighton*.

After loading this define statement, you can treat SCIENTISTS as a global variable whose value is the database list.

Important Notes

- You may not use iteration (e.g. do or loop) or assignment statements (e.g. no functions that end with '!'; all define statements must define functions, with the exception of the SCIENTISTS database definition). In short, you must follow pure functional programming style. Furthermore, you may not use any let expressions.
- A sample database definition is given in the define statement above, and a larger one is posted on the course web page. You may use this in developing and testing your code, but the code you turn in should not include a define statement for the database. This example is for your use during code development; we will run your submitted code on a different database, but (obviously) one with the same encoding as described above.
- The lists in your answers (when you use the sample database) must be sorted the same with the given examples. The same answers with different orderings are not valid.
- With the description of each function below are some sample calls to the function to illustrate what is intended by the textual description. These are not intended to be a complete test suite. You should thoroughly test your code by devising your own tests.
- The sample queries are done on the example database which is on the course webpage. So they are not running on the smaller sample database given in this document.

Your Tasks

1. Simple Queries

Write the following functions:

- `UNIVERSITY`: takes a single argument (a person name) and returns the university where the person took the PhD.
- `YEAR`: takes a single argument (a person name) and returns the graduation year of the person.
- `SUPERVISED-STUDENTS`: takes a single argument (a person name) and returns a list involving the supervised students of the person.

```
> (UNIVERSITY 'alan_turing)
'princeton_university
> (UNIVERSITY 'veysi_isler)
'bilkent_university
> (UNIVERSITY 'r_t_erdogan)
empty
```

```
> (YEAR 'john_kemeny)
1949
> (YEAR 'someone_not_in_database)
empty
```

```
> (SUPERVISED-STUDENTS 'steve_hedetniemi)
(list 'curtis_cook 'eleanor_hare 'sandra_mitchell)
> (SUPERVISED-STUDENTS 'alan_turing)
(list 'robin_gandy)
> (SUPERVISED-STUDENTS 'paul_taylor)
empty
> (SUPERVISED-STUDENTS 'someone_not_in_database)
empty
```

2. Where and When

Write the following functions:

- `UNIVERSITY-GRADUATES`: takes a single argument (a university) and returns a list involving the people graduated from that university.
- `YEAR-GRADUATES`: takes a single argument (a year) and returns a list involving the people graduated in that year.

```

> (UNIVERSITY-GRADUATES 'harvard_university)
(list 'doug_tygar)
> (UNIVERSITY-GRADUATES 'carnegie_mellon_university)
(list 'allan_heydon 'steve_quatterry 'shang_hua_teng)
> (UNIVERSITY-GRADUATES 'bogazici_university)
empty

```

```

> (YEAR-GRADUATES 1989)
(list 'niall_graham 'eleanor_hare 'paul_taylor)
> (YEAR-GRADUATES 1990)
empty
> (YEAR-GRADUATES 1991)
(list 'shang_hua_teng)

```

3. Supervisors

Write the `SUPERVISORS` function which takes a single argument (a person) and returns a list of professors who supervised the PhD thesis of this person.

```

> (SUPERVISORS 'w_e_singletary)
(list 'william_boone)
> (SUPERVISORS 'dina_kravets)
(list 'alok_aggarwal 'tom_leighton)
> (SUPERVISORS 'oswald_veblen)
empty
> (SUPERVISORS 'someone_not_in_database)
empty

```

4. Grand- Students and Super-Supervisors

Write the following functions:

- `GRAND-STUDENTS`: takes a single argument (a person) and returns a list involving the people who are supervised by the students of this person and by the students of students of this person, and etc. Observe that this is a recursively generated student list but it excludes the own students of the given person.
- `SUPER-SUPERVISORS`: takes a single argument (a person) and returns a list involving the people who are the supervisors of supervisors of this person and their supervisors, and etc. Observe that this is similarly a recursively generated supervisor list but it excludes the own supervisors of the given person.

```

> (GRAND-STUDENTS 'oswald_veblen)
(list 'william_boone 'george_sacerdote_e_singletary 'ann_yasuhara 'alfred_foster 'frank_harary
'lw_beineke 'niall_graham 'steve_hedetniemi 'curtis_cook 'eleanor_hare 'sandra_mitchell
'paul_stockmeyer 'john_kemeny 'bob_ritchie 'dick_hamlet 'scott_huddleston 'fred_springsteel
'stephen_kleene 'dana_scott 'michael_fourman 'john_longley 'kenneth_kunen 'alan_turing
'robin_gandy 'martin_hyland 'paul_taylor 'michael_o_rabin 'doug_tygar 'allan_heydon
'gary_miller 'hillel_gazit 'steve_quatterry 'susan_landau 'tom_leighton 'thang_bui
'dina_kravets 'shang_hua_teng)
> (GRAND-STUDENTS 'alan_turing)
(list 'martin_hyland 'paul_taylor)
> (GRAND-STUDENTS 'doug_tygar)
empty
> (GRAND-STUDENTS 'someone_not_in_database)
empty

```

```

> (SUPER-SUPERVISORS 'dina_kravets)
(list 'joseph_o_rourke 'norman_i_badler 'gary_miller 'michael_o_rabin 'alonzo_church
'oswald_veblen)
> (SUPER-SUPERVISORS 'curtis_cook)
(list 'alfred_foster 'alonzo_church 'oswald_veblen)
> (SUPER-SUPERVISORS 'joseph_o_rourke)
empty
> (SUPER-SUPERVISORS 'someone_not_in_database)
empty

```

5. Doctors of Era

Write the `IN-BETWEEN` function that takes two arguments (start year and end year) and returns a list of people who graduated in that time interval (**inclusively**).

```

> (IN-BETWEEN 1990 2000)
(list 'john_longley 'allan_heydon 'dina_kravets 'veysi_isler 'steve_guattery 'shang_hua_teng)
> (IN-BETWEEN 1800 1850)
(list 'pafnuty_lvovich_chebyshev)
> (IN-BETWEEN 1957 1957)
(list 'michael_o_rabin)
> (IN-BETWEEN 1700 1800)
empty
> (IN-BETWEEN 2000 1990)
empty

```

6. Flow of the Knowledge

Write the `FROM-TO` function that takes two arguments (two people) and returns a list of people where each successive elements of the list are a supervisor and his/her student. If there's no knowledge flow (i.e.: the first person is not a supervisor nor supervisor of the second) then empty list is returned. If the given arguments are the same person then a list involving only this given person is returned. The given people are also included in the list if there is an information flow.

Recall that there could be students with more than one supervisor. To ease your problem, you are going to choose the first supervisor defined in the database (i.e.: the uppermost one in the `SCIENTISTS` database). So the database can be treated as a tree for this specific question.

```

> (FROM-TO 'oswald_veblen 'dick_hamlet)
(list 'oswald_veblen 'alonzo_church 'john_kemeny 'bob_ritchie 'dick_hamlet)
> (FROM-TO 'alfred_foster 'frank_harary)
(list 'alfred_foster 'frank_harary)
> (FROM-TO 'steve_hedetniemi 'steve_hedetniemi)
(list 'steve_hedetniemi)
> (FROM-TO 'alan_turing 'veysi_isler)
empty
> (FROM-TO 'michael_o_rabin 'steve_guattery)
(list 'michael_o_rabin 'doug_tygar 'steve_guattery)

```

7. Common Supervisors

Write the `COMMON-SUPERVISORS` function that takes a list argument (people) and returns a list of people who are the common supervisors or super-supervisors of the people in the list.

```

> (COMMON-SUPERVISORS (list 'michael_fourman 'fred_springsteel 'dina_kravets))
(list 'alonzo_church 'oswald_veblen)
> (COMMON-SUPERVISORS (list 'michael_fourman 'fred_springsteel 'dina_kravets))
(list 'alonzo_church 'oswald_veblen)
> (COMMON-SUPERVISORS (list 'kenneth_kunen))
(list 'dana_scott 'alonzo_church 'oswald_veblen)
> (COMMON-SUPERVISORS (list 'tom_leighton 'andrei_andreevich_markov))
Empty

```

Submission

Your Scheme program should operate on a database of advances as illustrated above. A complete sample database called `scientists_database.scm` is on the website; you can use this database for testing your program.

Put all definitions of the functions above, as well as any other functions, in a file called `scientists_solution.scm`. Do not include the define statement for the SCIENTISTS database in the file you submit.

Your assignment will be marked not only for correctness, but for functional programming style, comments, and your testing strategy as well. Furthermore, we will test the code that you submit electronically on our own test cases, using a new SCIENTISTS database. For that reason, you must use the exact function names and argument lists that we have specified, and use the exact file name and submit instructions given above.

Notes

- Scheme program (DrScheme) with some manuals and tutorials are on the Download Files part of the course website.
- You're expected to run your code in Advanced Student Mode. (You can change this from the language section in DrScheme)
- In addition to the program source code and the program document, one test case (other than `scientists_database.scm`) must be submitted as a separate file. Prepare an additional report with this sample database of scientists you prepared, the inputs you used for testing, and the outputs must be included.
- The project will be done individually, not as a group.
- The deadline is 13.05.2008 at 00:00 for electronic submission and 14.05.2008 at 17:00 for hard copy submission.
- **Important!!!: Read the Programming Projects section in the document General Information for Students link in the Syllabus. You should obey the rules for electronic submission (e-mail subject, etc.) and documentation.**