Assignment 1: Comparing Languages

COMP 3010 – Organization of Programming Languages
January 2020

[**Note:** *This assignment is adapted from the original version authored by Prof. Michael Scott.*]

Problem description

In this assignment, you will solve a simple problem in each of five different programming languages:

- Ada
- OCaml or Haskell
- C#, Go, Rust, or Swift
- Python or Ruby
- Prolog

Each of your programs will output the Cartesian product of two or more sets. Implementations of Cartesian products in a representative set of programming languages are available on Rosettacode.org.

Your programs may either ask for the sets as input values, or "hard-code" the values of the sets. At a minimum, your code must demonstrate correct operation for the sets required in the task statement on Rosetta Code.

The representation for sets will depend on the implementation language. Some languages have built-in set types, or library classes/modules. In other languages, you may find it convenient to use lists, tuples, or arrays to represent sets. Another aspect in which languages will differ is support for Cartesian products of sets with different numbers of elements and/or elements of different types: e.g.,

```
{1,2,3}x{'a','b'} = {(1,'a'),(1,'b'),(2,'a'),(2,'b'),(3,'a'),(3,'b')}
```

If you already knew all the languages, you'd probably find your task easiest in Prolog or OCaml, and hardest in Ada, with the various other languages ranging in between. (Of course, you probably don't know all the languages already, so the unfamiliar ones will be the hardest.)

Hint: The required textbook's Companion Site provides a Zip file containing working versions of most of the nontrivial examples in the textbook. For Ada and C#, you might find it helpful to start with one of the examples: It will already import appropriate libraries and contain examples of the control constructs, I/O calls, etc.

At least one of your programs must be your own original code. For your other four programs, you may write your own code, or use solutions from Rosetta Code or another online source**.** Sadly, Rosetta Code does not include Ada or Prolog solutions. Look here for several solutions in Prolog; for Ada, consider using the Modula-2 solution at Rosetta Code as a model. **If you do use any online code in a program, you must provide a reference to that code (including a link) and fully comment the code to demonstrate that you understand how it works.** (Use of any online code without a reference will be considered a violation of the Academic Integrity policy.)

Division of labor and submission procedure

You may work alone on this project, or in teams of two or three students. If you split up the languages, whoever takes Ada should probably do fewer languages. (For example, in a two-person group, the person doing Ada might do two languages; the other person should do three.)

In any case, ***each team member must write his or her own README file (no sharing of text on this allowed), and turn in the project separately*** (with all five programs, which will be the same as the partner's code). This means, of course, that you'll need to really understand your partner's code.

Be sure your write-up (README file) describes any features of your code that the TA might not immediately notice. In addition, for this assignment, your README file must compare and contrast the programming experience in the different languages you used (all five of them):

- What was easy?
- What was hard?
- Are there noticeable differences in speed?
- What do you like/dislike?
- For any of your programs that use memoization, was this easy or hard to do in that language?

To turn in your code, use the following procedure, which will be the same for all assignments this semester:

1. Your code for all five programs should be in a directory called "`<YourName>_OPL_A1`" (for example, "`TomWilkes_OPL_A1`"). Put your write-up in a `README.txt` or `README.pdf` file in the same directory as your code.
2. In the parent directory of your A1 directory, create a `.tar.gz` file or a `.zip` file that contains your A1 directory (e.g., "`TomWilkes_OPL_A1.tar.gz`").
3. In the page for Assignment 1 on Blackboard, use the Submit button to upload your `.tar.gz` or `.zip` file. When you submit, give the name of your partner(s) (if any) in the submission comments field.

Resources

We will be using the following language implementations. All of these are available pre-installed on `cs.uml.edu`. If you encounter problems with any of these tools, please contact a CS lab assistant.

Ada

- Compile your code with `gnatmake` (a wrapper for the GNU Ada translator). It produces native executables.

C#

- Compile your code with `mcs` (the Mono project C# compiler) and run with the `mono` JIT/run-time system.

Go

- Run your code from the command line with `go`.

Haskell

- Run your code under the `ghci` interpreter, or compile with `ghc` (the Glasgow Haskell Compiler) to produce native binaries.

Prolog

- Run your code under the `swipl` interpreter.

Python

- Run your code under the `python3` interpreter.

OCaml

- Run your code under the `ocaml` interpreter, or compile with `ocamlc` to produce native binaries.

Ruby

- Run your code under the `ruby` interpreter.

Rust

- Compile your code with `rustc`.

Swift

- Run your code under the `swift` interpreter, or compile with `swiftc`.

We won't be devoting lecture time to how to use these languages. You'll need to find on-line tutorials or other resources and teach yourself. Here are some decent starting points:

Ada

- www.adacore.com
  Principal web repository for everything Ada, including documentation, compilers, and tools.
- www.adahome.com
  An older repository, now going somewhat stale, but still with some excellent content.
- www.adaic.org/ada-resources/standards/ada12/
  The Ada 2012 Reference Manual. (Note: This is a tough slog. You'll want to find a gentler tutorial—do a Google search.)

C#

- www.hitmill.com/programming/dotNET/csharp.html
  There isn't a canonical single site for C#, but this one is pretty good.
- www.ecma-international.org/publications/standards/Ecma-334.htm
  The international language standard. Covers C# 2.0.
- www.mono-project.com/
  The C# implementation we are using. Open source; supported largely by Xamarin, with a particular emphasis on high quality compilation of C# for the x86.
- msdn.microsoft.com/en-us/vstudio/hh341490.aspx
  Microsoft's main page for C# resources.

Go

- go-lang.org/
  Principal web site for Go. Includes a nifty in-the-web-page interpreter with which to experiment.

Haskell

- haskell.org/
  Principal web site for Haskell, with documentation, tutorials, implementations, tools, etc.
- Hoogle <https://www.haskell.org/hoogle/>
  Type or name based function search through all the library documentation.
- Learn You a Haskell for Great Good! <http://learnyouahaskell.com/chapters/>
  An easy-to-follow online book.

Prolog

- www.swi-prolog.org/
  The Prolog implementation we're using. Includes documentation and downloads (if you want to install it on your personal machine).

Python

- www.python.org/
  Principal web site for Python, with documentation, tutorials, implementations, tools, and news from the user community.

OCaml

- ocaml.org/
  Principal web site for OCaml, with documentation, tutorials, implementations, tools, etc.
- dev.realworldocaml.org/
  A free online version of a textbook on OCaml.

Ruby

- ruby-lang.org/
  Principal web site for Ruby, with documentation, tutorials, implementations, tools, etc.

Rust

- rust-lang.org/
  Principal web site for Rust, with documentation, tutorials, implementations, tools, etc.

Swift

- developer.apple.com/swift/
  Apple's official page for Swift.