CS403: Programming Languages

The Designer Programming Language

Preliminary information

Your task is to build an interpreter for a general purpose programming language of your own design. Your language must support the following features:

- comments
- integers and strings
- dynamically typed (like Scheme and Python)
- dictionaries with $O(\log n)$ worst-case access time
- arrays with O(1) access time
- conditionals
- recursion
- iteration
- convenient means to print to the console
- an adequate set of operators
- anonymous functions
- functions as first-class objects (i.e. functions can be manipulated as in Scheme - e.g. local functions)
- (graduate only) delayed evaluation of function arguments

The only basic types you need to provide are integer and string and you do not need to provide methods for coercing one type to another (although you may find it convenient to do so). The efficiency of your interpreter is not particularly important, as long as you can solve the test problem in a reasonable amount of time. Your language also does not need to support reclamation of memory that is no longer needed. You are to write your program in an imperative language such as C, C++, Python, or Java. Check with me first if you wish to use some other host language. **Your dictionary code must be written in your designer language.**

Test Problem

With your language, you are to simulate a one-bit full adder in a fashion similar to that in the book. The adder, logic gates, wires, and priority queue must be implemented as dispatch functions or objects. Your application need not be interactive. You can set the inputs programmatically at the beginning of the simulation and simply report the state of the output wire. Provide a README file that explains how to run the simulation and how to change the inputs.

Grading

- [100 points] everything works
- [51-99 points] functionality is missing/test program is missing
- [50 points] pretty printing
- [30 points] recognizing

Extra credit will be given to exceptional implementations. Your README file should give pertinent details.

For Undergraduates: if you do not, at least, implement a recognizer for your language, you will fail the course.

For Graduates: if you do not, at least, implement an pretty printer, you will fail the course.

Submitting the assignment

To submit your designer programming language, place all your source code, sample programs, a README detailing how to run and write programs in your language, and a makefile for building your system into one directory. Name the README file *README*. Your makefile should respond the command

```
make
```

which builds your processor and to the following commands, each of which illustrates a feature of your language:

```
make cat-error1
make run-error1
make cat-error2
make run-error2
make cat-error3
make run-error3
make cat-arrays
make run-arrays
make cat-conditionals
make run-conditionals
make cat-recursion
make run-recursion
make cat-iteration
make run-iteration
make cat-functions
make run-functions      # shows that functions are 1st-class
make cat-dictionary
make run-dictionary     # shows that you have a log(n) dictionary
```

The *cat* rules should print out the appropriate input program, while the *run* rules should execute the appropriate input program. In particular, the error rules should show off your parser detecting three different kinds of syntax errors.

Your makefile should also respond to the commands:

```
make cat-problem
make run-problem
```

These commands display the test problem of your implementation and run the test problem, respectively.

Note: in the case of recognizing only, only the error rules need be present in your makefile. In the case of pretty printing only, the *run* rules should run the input program through the pretty printer.

Finally, your makefile should respond to the command:

```
make clean
```

This command should remove all compilation artifacts (such as `.o` or `.pyc` files) so that a clean compile can be performed.

Makefiles for graduate students should additionally respond to the commands:

```
make cat-grad
make run-grad
```

To submit your language interpreter, run the command:

```
submit proglan lusth dpl
```

Demonstrating your assignment

You may demonstrate your programming language within one week after the due date. Send me email if you desire to do so. Demonstration is not required, but your may wish to show off your creation or beg for more points.

There are only a limited number of slots available for demonstrations. Not everyone will be able to demo.