CS4215 Programming Language Implementation

Lab task for Week 12-13
A Domain-Specific Language for Shell Scripting

This mini-project must be completed by 6pm on the 14th Apr 2015. You are to submit your system and report (not exceeding 5 pages) in a zip file into IVLE, give a short 10-min demo of your implementation.

Domain-specific languages are languages that are customized to some specific purposes. They allow domain experts to built their systems in a faster and more convenient way. This mini-project is expected to be completed within a 2-weeks period. We provide one example here, but you are free to propose your own domain-specific language as your project. The only requirement is that it be developed using the OCaml compilation infrastructure (such as camlp4), which we have been using for our course.

1 Shell Scripting

Shell scripts are a quick way to write some file processing applications over the Unix-based operating system. As a simple example, the following shell script would list the size characteristics of the OCaml programs in the current directory.

```
echo "################"
echo "INITIAL SOLUTION"
files="*.ml"
wc ${files}
```

Its output on our directory is:

```
INITIAL SOLUTION
 22  88 623 debug_init.ml
 1369 10299 52920 debug.ml
 104  396 3338 error.ml
 2149 10838 65046 gen.ml
 1746  7568 46056 globals.ml
 43 134 708 pr0.ml
 42 132 692 sh_proc.ml
 118  437 3172 VarGen.ml
```

We have also provided four other Shell scripts for you to use as simple examples. Please execute them to see what they are doing.

2 Direct Implementation in OCaml (40%)

Your first task is to convert each of these shell scripts into the simplest possible OCaml program that would achieve the same effect. If a feature, such as variable, is already present in OCaml, we suggest that you use it where possible. This way, your OCaml scripting program will have much better control and flexibility. For the above simple example, we can use the following OCaml program to execute the same task as `ws0.sh`.

```
let exec_imm s =
  let code = Sys.command s in
  code;;
let echo s =
  print_endline s;;
let () = echo "################";;
let () = echo "INITIAL SOLUTION";;
let files = "*.ml";;
let c = exec_imm ("wc "^files);;
```

3 A DSL Extension for Scripting (60%)

Your next task is to design and implement a language extension for OCaml that would facilitate Shell scripting. In general, there are many ways to achieve this. I would therefore like you to consider simple and effective solutions that are familiar, so that it is easy for programmers to learn how to use this new language extension.
In the case of Shell scripting, the commands may either result in a string output or be directly executed. Thus, we could envision two annotation to support such scripting. For example, instead of using:

```
let c = exec_cmd ("wc "^files);;
```

one may use:

```
let c = @cmd{wc ${files}};;
```

to indicate that the text `wc ${files}` is a scripting command which is to be executed and would return an integer status code.
If we had expected a text file to be returned from the expression, we may use a different command, say:

```
let c = @out{wc ${files}};;
```

This would return a list of strings as output from the scripting command. These two simple ideas are how we may support shell scripting in OCaml. Please look through the other scripting examples, and design your own solution to this OCaml-based scripting language extension. Due to its flexibility, we suggest for you to use `camlp4` for this language extension.