

Project 3

CptS 355

Spring 2005

21st March 2005

Assigned: 21 March, 2005

Due: Friday, April 8, 2005, 11:59:59PM. Develop all your code in a directory named “three”. When you are finished make a gzipped tar file or zip file of the directory and turn it in using the turn-in page as for the first assignment. Include a README file describing the environment for building and running your interpreter and containing complete directions on how to build the interpreter from your source files. The name of the executable file should be `sps.exe` or just `sps`.

Credit: This assignment will count approximately 12% of your final grade.

Policy: This assignment is to be your own, personal, individual work. Do not work with other students on it: if you have a question or are stumped see the instructor or the TA.

Project 3 is intended to enhance understanding of the behavior of, and mechanisms used to implement, static and dynamic scope rules.

You will be modifying your SPS interpreter to handle a slightly different language. Let’s call it Scoped Simple PostScript, SSPS. SSPS has no `dict` operation. Instead, each time a procedure is called a new dictionary is automatically pushed on the dictionary stack. The dictionary must be able to hold at least 20 names (of course you can use an implementation that handles an arbitrary number of names).

The SPSS interpreter will take a command line switch, either `-s` or `-d`, to indicate whether it should behave using static scope rules or dynamic scope rules. The default if no switch is supplied is dynamic scope rules. *Hint:* use a procedure variable for the name lookup function. That way your mainline code will not have to choose static or dynamic behavior for each name lookup. Instead the choice can be made once when the program starts.

Using dynamic scope rules SPSS will behave very much like SPS except that there is no need to use `dict` operations in programs. My interpreter for dynamic SSPS was

actually a little smaller than my original SPS interpreter because I no longer had to detect the `dict`, `begin`, and `end` operations. Instead, each time I was about to call a procedure I unconditionally pushed a new dictionary and when a procedure was about to return I popped the dictionary.

To implement static scope rules you need a *static chain*. I suggest keeping a pointer to the head dictionary of the static chain in a variable at all times. If that variable is called `scp` then at the point where a procedure is defined (*i.e.*, when doing a `def` for a block of code) you need to save the current value of `scp` as part of the definition of the procedure. When calling a procedure you will create a new dictionary automatically (as in the dynamic case) but in addition to pushing it on the dictionary stack you need to initialize a new field of the dictionary structure (it might be called `staticLink`) to contain the link saved as part of the procedure's definition. *Hint*: in my implementation this all took only a handful of lines of new code but it was tricky to get all the pieces right. My advice is think more, write less for this part of the project.

As discussed in class, variable lookups now proceed by looking in the current dictionary at the top of the dictionary stack and then following the `staticLink` fields to other dictionaries (instead of just looking at all the dictionaries on the stack in turn, which give dynamic scope rules).

Note on interpreter input

Please arrange things so that your interpreter reads its SSPS input from the *standard input*.

That is, it must be possible to run your interpreter on file `input.ps` by executing

```
ssps.exe <input.ps          (for windows)
ssps <input.ps              (for Unix/Linux)
```

If all the interpreters handle input the same way it makes things easier for us when we evaluate your work.

Output of the interpreter

Note that many people did not produce the required output in Project 2. You lost some points for this. You will lose more points this time.

The output of the SPSS interpreter consists of the contents of the stack when the end of the program is reached, printed one value per line **and the contents of the dictionary stack printed one name and value per line**. Put a line containing "----" between different dictionaries on the dictionary stack. (Note the new requirement to print the contents of the dictionary stack.)

What if my SPS interpreter didn't work correctly?

You will need to complete it for this project as I stated before break. Some people didn't get the procedure calling feature of SPS working in Project 2. Since this project is worth more and re-uses the sps interpreter code it is a way to make up so ground on your grade.

There is not much point in doing this project if procedure calling doesn't work, so *ask for help* from Tammy or me. As usual, this project is to be **your personal, individual work**.

How can I tell if my static scoping code is working correctly?

You will have to create some test cases for which you can predict the correct answers. We covered one simple example in class. Translated to SPS and simplified a bit that example looks like:

```
/x 4 def
/g { x } def
/f { /x 7 def g } def
f
```

which will leave 7 on the stack using dynamic scoping and 4 using static scoping.