CSL302 : Assignment 5

April 30, 2007

Contents

1 Objective

To understand how stack frames are maintained when the language allows for nesting of procedures.

2 Assignment Statement

A new programming language called Gaul is being designed. The designers of this language have allowed for nesting of procedures, static scoping, pass by value parameter passing, recursion. You have an opportunity for a Summer Internship with the Gaul project provided you can impress the designers by doing the following tasks.

1. Design a stack frame layout for procedures in Gaul.

2. Make a menu driven program which can perform the following

   MENU ITEM1 Show all procedures that can be called from within a specified procedure.

   MENU ITEM2 Call one of those procedures.

   MENU ITEM3 Return from the procedure.

   MENU ITEM4 Show all visible variables(local and global) and their values, from within the current procedure.

   MENU ITEM5 Set a variable to an expression.(eg. y = x+5)

   MENU ITEM6 Display all the fields of the stackframe in a Readable Format.

3 Testcases

1. Use the following testcases. Convert it into you internal format and put it along with the submission.

3.1 TESTCASE 1

```
program P()
{
        var gauls, romans ;
        assign gauls = 0, romans = 0 ;
        procedure P1(params x1,y1)
        {
                var asterix, obelix, vitalstatistix ;
                assign asterix = 5 , obelix = 4 , vitalstatistix = 5 , gauls = 1 ;

                procedure P11(params x11)
                {
                        var cacafonix, fulliautomatix ;
                        assign cacafonix = 2, fulliautomatix = 1 ;
                }

                procedure P12(params x,z)
                {
                        var asterix, caesar, cleopatra ;
                        assign asterix = 0 , caesar = 3, cleopatra = 4, romans = 1 ;
                        procedure P121(params x)
                        {
                                var geriatrix ;
                                assign geriatrix = 0, cleoaptra = 1 , asterix = 10 ;
                        }
                }

        }

        procedure P2(params y2,z2)
        {
                var caesar, cleopatra ;
                assign caeser = 2, cleopatra = 1, romans = 1 ;

                procedure P21(params x21,y21)
                {
                        var brutus, cassius ;
                        assign brutus = 1, cassius = 1 ;
                        procedure P211(params x2,z2)
                        {
                                assign cleopatra = 2 ;
                        }

                        procedure P212(params x2)
                        {
                                var caesar ;
                                assign caesar = 1000 ;
                                procedure P2121()
                                {
                                        var brutus, cassius ;
                                        assign brutus = 1, cassius = 1 ;
                                }
                        }


                        procedure P213(params x213)
                        {
                                var octavius, lepidus, antony ;
                                assign octavius = 1, lepidus = 3, antony = 4,
                                caesar = 0, gauls = 1 ;
                                procedure P2131(params y2)
                                {
                                        var augustus ;
                                        assign augustus = 1, brutus = 0, cassius = 0 ;
                                }
                        }
                }
        }
}
```

3.2 TESTCASE 2

```
program P()
{
        var imperative, functional ;
        assign imperative = 0, functional = 0 ;

        procedure P1(params y,z)
        {
                var ml, lisp, haskell ;
                assign functional = 111, lisp = 112, haskell = 113, ml = 114 ;

                procedure P11(params x,y)
                {
                        var scheme, matlab ;
                        assign scheme = 121, matlab = 122 ;
                        procedure P111(params z)
                        {
                                var erlang, lisp ;
                                assign erlang = 131, haskell = 132, lisp = 133 ;
                        }

                        procedure P112(params x)
                        {
                                var scheme, matlab, pascal ;
                                assign scheme = 141, matlab = 142, pascal = 143 ;
                                procedure P1121(params z)
                                {
                                        var ml, lisp ;
                                        assign ml = 151, lisp = 152 ;
                                }

                                procedure P1122(params x,y)
                                {
                                        var java ;
                                        assign java = 151, matlab = 152 ;
                                        procedure P11221(params x,z)
                                        {
                                                var erlang, lisp ;
                                                assign erlang = 161, haskell = 162,
                                                       lisp = 163 ;
                                        }
                                }
                        }
                }
        }

        procedure P2(params y,z)
        {
                var c, java, pascal;
                assign imperative = 211, c = 212, java = 213, pascal = 214 ;

                procedure P21(params x,y)
                {
                        var scheme, matlab ;
                        assign scheme = 221, matlab = 222, pascal = 223 ;

                        procedure P211(params x,z)
                        {
                                var ml, lisp ;
                                assign functional = 231, ml = 232, lisp = 233 ;
                                procedure P2111(params z)
```

```
                                {
                                        var ml, lisp ;
                                        assign functional = 241, ml = 242, scheme = 243,
                                               z = 244 ;
                                }
                        }

                        procedure P212(params x,y)
                        {
                                var java ;
                                assign java = 231, matlab = 232 ;
                                procedure P2121(params x,z)
                                {
                                        var erlang, lisp ;
                                        assign erlang = 241, c = 242, lisp = 243  ;
                                }
                        }
                }
        }
}
```

3.3 TESTCASE 3

```
program P()
{
        var logicians, mathematicians, scientists ;
        assign logicians = 0, scientists = 0 ;
        procedure P1(params x)
        {
                var godel, church, turing ;
                assign logicians = 111, godel = 112, church = 113, turing = 114 ;
                procedure P11(params godel,church)
                {
                        var skolem, morgan ;
                        assign skolem = 121, morgan = 122, godel = 123 ;
                        procedure P111(params godel,church)
                        {
                                var aristotle, plato ;
                                assign aristotle = 131, plato = 132, godel = 133, x = 911 ;
                        }
                }
        }

        procedure P2(params x,y)
        {
                var einstein ;
                assign scientists = 21, einstein = 22, x =22 , y = 23 ;
                procedure P21(params x,y)
                {
                        var heisenberg, planck ;
                        assign heisenberg = 211, planck = 212, y = 213 ;
                        procedure P211(params x)
                        {
                                var fermat, riemann ;
                                assign fermat = 221, riemann = 222, x = 223 ;

                                procedure P2111(params y)
                                {
                                        var skolem, morgan;
                                        assign skolem = 231, morgan = 232, x = 911 ;
                                        procedure P21111(params x)
                                        {
                                                var aristotle, plato ;
                                                assign aristotle = 241, plato = 242,
                                                       x = 243, y = 244 ;
                                        }
                                }
                        }
                }
        }
}
```

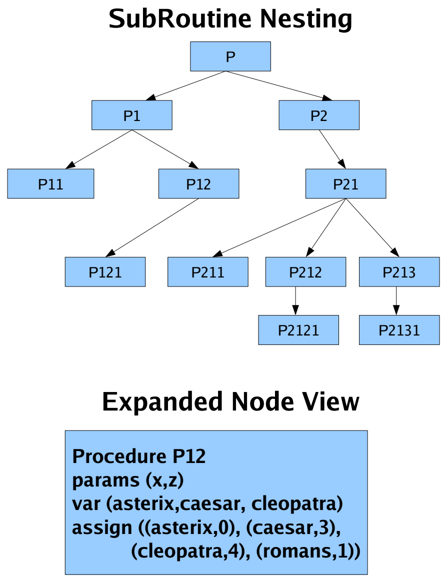
4 INPUT FORMAT

You can encode the above given testcase in a suitable representation and use it as input for your program. Your program should work for any input in this representation. A sample graph based representation given in the next page. This graph representation encodes the nesting of subroutines/procedures. Add the following information to each node in the graph.(As shown in the expanded node view)

1. The list of parameters.
2. The list of variable declarations.
3. The list of assignments.

5 NOTE

1. You don't need to parse the input. You can manually convert it to your internal representation.



```
Procedure P12
params (x,z)
var (asterix,caesar, cleopatra)
assign ((asterix,0), (caesar,3),
        (cleopatra,4), (romans,1))
```

Figure 1: Graph Representation of TestCase1

2. The variable names don't signify anything.
3. Assume all variables to be of type Integer.

Programming Language Java/SML

Submission Deadline 11.55 p.m. , 30th April 2007(Monday).