

# Homework 4

**Due:** The deadline was extended until **Tuesday, August 7, 1 pm, for the Day section**, and **Wednesday, August 8, 6 pm, for the Evening section**, in class or in 324 drop box. Because of the extension, no homeworks are taken later than this.

**Note:** Print this file and hand in the printout after filling in the answers to the questions. No electronic submission this time. Please use an unsealed envelope, having on top the cover page provided at the end of this file.

**Note:** For this homework, you are not allowed to discuss your ideas with your colleagues.

## 1 Question 1

Consider the following program in an imperative language:

```
main() {
    int i;
    int a[4];

    void p(int one, int two)
        { one = two;
          two = 4;
        }

    int f(int j)
        { a[j]=a[j]+1;
          return j+1;
        }

    i=0;
    a[0]=1; a[1]=2; a[2]=3; a[3]=4;

    p(i, a[i]);
    write(i,a[0],a[1],a[2],a[3]);    /* prints values */
                                    /* on a new line */

    p(a[f(i)], a[i]);
    write(i,a[0],a[1],a[2],a[3]);    /* prints values */
                                    /* on a new line */
}
```

NOTE:

- Assume that for call-by-value, initial values of formal parameters are copied sequentially (left to right) from the actual parameters.

- Assume that for call-by-value-result, initial values of formals are copied sequentially (left to right) from the actual parameters; final values of formals are copied sequentially (left to right) AT RETURN TIME into the actual parameters. (i.e., the location to copy to is determined at return time.)
- Assume that for call-by-reference, the location to be pointed to by the formals is determined sequentially (left to right) through the actuals at the time of the procedure call.

Give the output of the program, assuming the parameters are passed:

a) by value;

b) by value-result;

c) by reference;

d) by name.

## 2 Question 2

Consider the following code in an imperative language:

```

program P;
  var a,b,c : integer;
  procedure R;
    var a : integer;
  begin {R}
    a := 4; b := 7; c := 5;
    write a,b,c;          <-- 2) ----- . a ----- . b ----- . c
  end; {R}
  procedure Q;
    var a, b : integer;
    procedure S;
      begin {S}
        write a,b,c;    <-- 1) ----- . a ----- . b ----- . c
        R;
      end; {S}
    begin {Q}
      a := 5; b := 1; c := 6;
      S;
    end {Q}
  begin {P}
    a := 1; b := 2; c := 3;
    Q;
  end; {P}

```

Assume the language uses **dynamic scope**.

a) What would the program print?

1)

2)

b) What is the scope of a, b, and c in the points marked by 1) and 2)? Fill in the dashed spaces above with the name of the procedure that particular use of the variable was declared in. Example: P . a

c) Draw the activation tree (showing what procedures call what procedures):

### 3 Question 3

Consider the following code in an imperative language:

```
program M;
  var x,y,z : integer;
  procedure A(flag:boolean);
    var x : integer;
    procedure B;
      var z : integer;
      begin {B}
        z := 5;
        A(false);
      end ; {B}
    begin {A}
      ...
      x := 7;
      if (flag) then B else C;
      ...
    end; {A}
  procedure C;
    var y : integer;
    procedure D;
      begin {D}
        write x,y,z;      <-- WHEN YOU GET HERE
      end; {D}
    begin {C}
      y := 1;
      D;
    end {C}
  begin {M}
    x := 1; y := 2; z := 3;
    A(true);
  end; {M}
```

a) Show what the runtime stack would look like the first time you reach the statement labeled "WHEN YOU GET HERE". Show the stack frames including control links, access links, formal parameters (and their actual values), local variables (and their values, i.e.  $x=5$ ), and procedure declarations. Fill in the blanks in the runtime stack on the next page. Note: The stack starts at the bottom and grows upwards.

-----  
params:  
local vars:  
local procs:  
control link:  
access link:  
-----

params:  
local vars:  
local proc:  
control link:  
access link:  
-----

params:  
local vars:  
local procs:  
control link:  
access link:  
-----

params:  
local vars:  
local procs:  
control link:  
access link:  
-----

params:  
local vars:  
local procs:  
control link:  
access link:  
-----

M  
params:  
local vars:  
local procs:  
control link:RTS  
access link: null  
-----

RTS  
-----

b) Assume the language uses lexical scope. What would the program print?

When you reach the statement labeled "WHEN YOU GET HERE", what is the scope of x, y, and z? Fill in the dashed space with the name of the procedure the variable was declared in (i.e, M . x) :

\_\_\_\_\_ . x                      \_\_\_\_\_ . y                      \_\_\_\_\_ . z

c) Assume the language uses dynamic scope. What would the program print?

When you reach the statement labeled "WHEN YOU GET HERE", what is the scope of x, y, and z? Fill in the dashed space with the name of the procedure the variable was declared in:

\_\_\_\_\_ . x                      \_\_\_\_\_ . y                      \_\_\_\_\_ . z

d) Draw the activation tree (showing what procedures call what procedures):

## Cover Page for Homework 4

Family name: \_\_\_\_\_ Student #: \_\_\_\_\_

First name: \_\_\_\_\_ CDF id: \_\_\_\_\_

Section:      Day section      Evening section

I declare that this assignment is my own work, and it is in accordance with the University of Toronto Code of Behaviour on Academic Matters and the Code of Student Conduct. I declare that I didn't discuss this assignment with anybody else.

Signature \_\_\_\_\_

The following is for our use in grading:

Q1 \_\_\_\_\_ / 20

Q2 \_\_\_\_\_ / 10

Q3 \_\_\_\_\_ / 20

\_\_\_\_\_

Total \_\_\_\_\_ / 50

Other (penalties)

Final mark \_\_\_\_\_ / 50