Written Exam - November 2010
(some elements of solutions)

These 3 exercises are independent, but all them are based on the “While” language (studied in the lecture course), and those syntax is recalled in Appendix A.

Exercise 1. Procedures

The operational semantics of non-recursive procedures (without) parameters for the “While” language is recalled in Appendix B. We consider the following program $P$:

```
begin
  var x:=1 ;
  var y:=3 ;
  proc p1 is x:=x+1 ;
  /* location l0 */
  proc p2 is
  begin
    var x:=4 ;
    proc p is y:= x ;
    /* location l1 */
    call p ;
    call p1 ;
    /* location l2 */
    end ;
begin
  var x:=2 ;
  proc p1 is x:=x+y ;
  proc p is y:=x+y ;
  /* location l3 */
  call p2
  /* location l4 */
end
```
Note that to properly answer this question you should follow the **execution order** of the program, namely 10, 13, 14, 11 and 12

**Q1.** We assume a dynamic binding for the variables and procedures. Give the values of the state $\sigma$ at each location 10 to 14 and of the procedure environment $Env_P$ at locations 11 and 13.

At 10:
\[ \sigma = \{ x \mapsto 1, y \mapsto 3 \} \]

At 11:
\[ \sigma = \{ x \mapsto 4, y \mapsto 3 \} \]
\[ Env_P = \{ p1 \mapsto x := x + y, \ p \mapsto y := x \ \ p2 \mapsto \text{begin...end} \} \]

At 12:
\[ \sigma = \{ x \mapsto 8, y \mapsto 4 \} \]

At 13:
\[ \sigma = \{ x \mapsto 2, y \mapsto 3 \} \]
\[ Env_P = \{ p2 \mapsto \text{begin...end}, \ p1 \mapsto x := x + y, \ p \mapsto y := x + y \} \]

At 14:
\[ \sigma = \{ x \mapsto 2, y \mapsto 4 \} \]

**Q2.** Same question when we assume a dynamic binding for the variables and a static binding for the procedures.

At 10:
\[ \sigma = \{ x \mapsto 1, y \mapsto 3 \} \]

At 11:
\[ \sigma = \{ x \mapsto 4, y \mapsto 3 \} \]
\[ Env_P^1 = \{ p1 \mapsto (x := x + 1, \emptyset) \} \text{ before declaration of p} \]
\[ Env_P = Env_P^1\{p \mapsto (y := x, Env_P^1)\} \text{ the answer} \]

At 12:
\[ \sigma = \{ x \mapsto 5, y \mapsto 4 \} \]

At 13:
\[ \sigma = \{ x \mapsto 2, y \mapsto 3 \} \]
\[ Env_P^1 = \{ p1 \mapsto (x := x + 1, \emptyset) \} \text{ external declaration of p1} \]
\[ Env_P^2 = Env_P^1\{p2 \mapsto (\text{begin...end}, Env_P^1)\} \text{ external declaration of p2} \]
\[ Env_P^3 = Env_P^2\{p1 \mapsto (x := x + y, Env_P^2)\} \text{ re-declaration of p1} \]
\[ Env_P = Env_P^3\{p \mapsto (y := x + y, Env_P^3)\} \text{ the answer} \]

At 14:
\[ \sigma = \{ x \mapsto 2, y \mapsto 4 \} \]
Exercise 1. An extension of the “While” language

We extend the “While” language by introducing a new statement whose syntax is the following:

\[
\text{for } x := a_1 \text{ to } a_2 \text{ do } S
\]

Q1. Give a “While” program \(P_2\) such that \(P_2\) is equivalent to \(P_1\) and \(P_2\) does not contain any “for” statement.

var x := 0;
begin
  var v1:=1 ; var v2:=100 ;
  var x:=v1 ;
  while x <= v2
    print (x) ;
    x := x + 1 ;
end ;
-- useless to restore x to its previous value,
-- since we used a nested block with a local x inside
print(x)

Q2. Give a semantic rules for the “for” statement:

1. Using the “while” statement
2. Without using the “while” statement.

1. A solution using the while statement:
   - If \(A[a_1] \sigma \leq A[a_2] \sigma\) then
     \[
     (S, \sigma[x \mapsto v_1]) \rightarrow \sigma' , \quad (\text{while } x < a_2 \text{ do } S , \sigma') \rightarrow \sigma''
     \]
     \[
     (\text{for } x := a_1 \text{ to } a_2 \text{ do } S, \sigma) \rightarrow \sigma''[x \mapsto \sigma(x)]
     \]
     where \(v_1 = A[a_1] \sigma\) and \(v_2 = A[a_2] \sigma\)

   - If \(A[a_1] \sigma > A[a_2] \sigma\) then
     \[
     (\text{for } x := a_1 \text{ to } a_2 \text{ do } S, \sigma) \rightarrow \sigma
     \]

2. A solution without using the while statement:
   - If \(A[a_1] \sigma \leq A[a_2] \sigma\) then
     \[
     (S, \sigma[x \mapsto v_1]) \rightarrow \sigma' , \quad (\text{for } x := \sigma'(x) \text{ to } a_2 \text{ do } S, \sigma') \rightarrow \sigma''
     \]
     \[
     (\text{for } x := a_1 \text{ to } a_2 \text{ do } S, \sigma) \rightarrow \sigma''[x \mapsto \sigma(x)]
     \]
     where \(v_1 = A[a_1] \sigma\) and \(v_2 = A[a_2] \sigma\)

   - If \(A[a_1] \sigma > A[a_2] \sigma\) then
     \[
     (\text{for } x := a_1 \text{ to } a_2 \text{ do } S, \sigma) \rightarrow \sigma
     \]

Q3. According to your rules, what are the values printed when executing the following program \(P_3\)?
The values printed are 0, 8, 9, 0.
Exercise 3. Code generation for the AM machine - 6 pts.

Q1. *Extend the function CS to deal with the new “While” statement repeat S until b*

\[ CS[\text{repeat } S \text{ until } b] = CS[S] : \text{LOOP}(CS[¬b], CS[S]) \]

Q2. *Give the code produced for the following statement:

\[ \begin{align*}
\text{repeat} \\
&\quad x := x+2 ; \\
&\quad y := y+1 ; \\
&\quad \text{until } x=y ;
\end{align*} \]

\[ \begin{align*}
\text{Push-2:Fetch}(x):\text{Add:Store}(x):\text{Push-1:Fetch}(y):\text{Add:Store}(y): \\
\text{Loop}(
&\quad \text{Fetch}(y):\text{Fetch}(x):\text{EQ:Neg}, \\
&\quad \text{Push-2:Fetch}(x):\text{Add:Store}(x):\text{Push-1:Fetch}(y):\text{Add:Store}(y): \\
\) \]