

## Programming Languages and Software Design PLSD 210

### Tutorial Exercises 1

Due: week 4

1. Write EBNF definition for:
  - a. C if statement
  - b. C switch statement
  - c. C procedure declaration
  
2. Draw syntax graphs (by hand or using LaTeX Rail) for:
  - a. C if statement
  - b. C switch statement
  - c. C procedure declaration
  
3. Consider the following grammar  
 $S \rightarrow A B C$   
 $A \rightarrow a A \mid a$   
 $B \rightarrow b B \mid b$   
 $C \rightarrow c C \mid c$ 
  - a. Rewrite the grammar in EBNF
  - b. Generate a derivation tree for start symbol S
  - c. What terminals does this grammar produce ?
  
4. Design a grammar in EBNF,  
which generates terminals in the form: ab, aabb, aaabbb, and so on  
That is, the number of heading "a"s has to match the number of trailing "b"s.

## Programming Languages and Software Design PLSD 210

### Tutorial Exercises 2

Due: week 6

Compute the weakest preconditions from the following statements with corresponding postconditions.

1.  $\{P\} \quad x := x + y \quad \{x > 0\}$
  
2.  $\{P\} \quad x := 2 * y + 1;$   
 $\quad \quad y := y - 3 \quad \{y < 0\}$
  
3.  $\{P\} \quad \text{IF } x < 0 \text{ THEN } x := -x \text{ END } \{x > 0\}$
  
4.  $\{P\} \quad x := 1;$   
 $\quad \quad \text{FOR } i := 1 \text{ TO } 10 \text{ DO } x := 2 * x \text{ END } \{x = 2^{10}\}$
  
5.  $\{P\} \quad x := 0;$   
 $\quad \quad \text{FOR } i := 1 \text{ TO } 10 \text{ DO } x := 2 * x \text{ END } \{x = 2^{10}\}$

## Programming Languages and Software Design PLSD 210

### Tutorial Exercises 3

Due: week 7

Define an abstract data type for a queue. New elements are inserted at the queue's head. Elements are read from the queue's tail.

Consider the following functions:

```
create
delete
is_empty (* test if empty *)
put_head (* insert new element *)
cut_tail (* delete oldest element *)
get_tail (* read oldest element without deleting *)
```

1. Define the properties of a queue
2. Write the declaration of the queue interface (in pseudo notation)
3. Write the implementation of the queue routines (in pseudo notation)
4. Show the usage of the queue ADT from an application program (in pseudo notation)

## Programming Languages and Software Design PLSD 210

### Tutorial Exercises 4

Due: week 9

```
MODULE m;

  PROCEDURE p0(): INTEGER;
  BEGIN RETURN 0
  END p0;

  PROCEDURE p1(a,b: INTEGER): INTEGER;
  VAR c: INTEGER;

      PROCEDURE p2(x: INTEGER): INTEGER;
      PROCEDURE p3(x: INTEGER): INTEGER;
      BEGIN (* p3 *)
        IF x = 1 THEN RETURN p0() ELSE RETURN p3(x DIV 2) + 1 END
      END p3;
      BEGIN (* p2 *)
        c := p3(b);
        RETURN a + c
      END p2;

  BEGIN (* p1 *)
    RETURN p2(a+b);
  END p1.

BEGIN (* main *)
  p1(1,2);
END m.
```

- A. Use an implementation with static and dynamic links  
Draw the *stack* contents at the time *p0* is called.
- B. Use an implementation with display and dynamic links  
Draw the *stack* and *display* contents at the time *p0* is called.

## Programming Languages and Software Design

### PLSD 210

#### Tutorial Exercises 5

Due: week 10

---

```
MODULE m;
VAR sum: REAL;

PROCEDURE average(f: PROC; lower, upper: REAL): REAL;
VAR i: INTEGER; sum: REAL;
BEGIN
  sum := 0.0;
  FOR i:=lower TO upper DO
    sum := sum + f(i)
  END;
  RETURN sum / (upper - lower + 1)
END average;

PROCEDURE f1(x: REAL): REAL;
BEGIN
  RETURN 2*x + sum
END f1;

PROCEDURE f2(x: REAL): REAL;
BEGIN
  RETURN x*x - sum
END f2;

BEGIN (* main *) sum := 7.0;
  WriteReal(average(f1, 0.0, 9.0);
  WriteReal(average(f2, 2.5, 3.5);
END m.
```

- A. Assuming **static** scoping: Label all variable references with proc-offset/local offset.
- B. Assuming **static** scoping: Use an implementation with static and dynamic links. Draw the *stack* contents at the time *f1* is called.
- C. Assuming **dynamic** scoping: Use an implementation with static and dynamic links. Draw the *stack* contents at the time *f2* is called.

## Programming Languages and Software Design

### PLSD 210

#### Tutorial Exercises 6

Due: week 12

---

The “Producer-Consumer Problem” is extended into the “Bounded Buffer Problem”, by providing more than just one buffer space.

Assume one producer process has to communicate with one consumer process. But now there are 10 buffer spaces available.

Implement the bounded buffer problem in C or pseudo language. Define two processes named “producer” and “consumer”. Use a ring FIFO buffer (first in first out = queue) to store packets. Be sure to have the right number and initialization of semaphores for securing the critical section.

# Programming Languages and Software Design

## PLSD 210

Tutorial Exercises 7

Due: week 13

---

Two large matrices  $A$  and  $B$  are to be added. Each matrix has  $1000 \times 1000$  elements.  
Example:

$$C = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Use pseudo-notation for the following problems.

- A. Implement the matrix addition for a sequential system.
- B. Implement the matrix addition for a MIMD parallel system using monitors.
- C. Implement the matrix addition for a distributed MIMD system using messages.
- D. Implement the matrix addition for a SIMD data parallel system.