

Principles of Programming Languages

Small examination

Student ID:

Name:

Problem 1 Illustrate the quilts represented by the following expressions (1), (2), and (3) in the language Little Quilt.

(1) `sew (turn (turn (b)), a)`

(2) `let`
 `val x = turn (b)`
 `in`
 `sew (x,x)`
 `end`

(3) `let`
 `fun unturn (x) = turn (turn (turn (x)))`
 `fun pile (x,y) = unturn (sew (turn (y), turn (x)))`
 `val aa = pile (a, turn (turn (a)))`
 `val bb = pile (unturn (b), turn (b))`
 `in`
 `sew (aa, bb)`
 `end`

The meaning of `a`, `b`, `turn`, `sew` are as follows. The other constructs of Little Quilt (`let` expressions, `val` declaration, `fun` declaration) have the meaning explained in the lecture.

- Expressions `a` and `b` represent the quilts in Figure 1 and Figure 2 respectively.

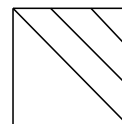
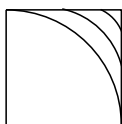


Figure 1: The quilt that `a` represents Figure 2: The quilt that `b` represents

- The expression `turn (e)` represents the quilt obtained by rotating 90 degrees to the right the quilt represented by the expression `e`.
- The expression `sew (e1, e2)` represents the quilt that is obtained by sewing the two quilts `e1` and `e2`, where `e1` is in the left side and `e2` is in the right side, and they must have the same height.

Problem 2 Answer the following problems about the control flow in the imperative language presented in the lecture.

(1) Illustrate the control flow of the following program fragment.

```
if x>0 then x := x - 1
else if y>0 then y := y - 1
     else y := y + 1
```

(2) Illustrate the control flow of the following program fragment.

```
while x>0 do
  begin
    if x=3 then
      begin
        x := x - 1;
        continue
      end;
    y := y + 1;
    x := x - 1
  end
```

Problem 3

Derive the Hoare triples (1), (2), and (3) by using the rules presented in the lecture.

(1) $\{a = 3\} a := a + 1 \{a = 4\}$

(2) $\{a = 3\} a := a + 1; a := a + 2 \{a = 6\}$

(3) $\{x = 5\} \mathbf{while} \ x > 0 \ \mathbf{do} \ x := x - 1 \ \{x = 0\}$

Problem 4

Show the output produced by executing the following Pascal program. When the keyword `var` is attached to a formal parameter, it designates the parameter as call-by-reference. The procedure `writeln` writes out to the standard output the value of the parameter and a new line character.

```
program test;                                begin
var x : integer;                             x := 3;
var y : integer;                             y := 4;
procedure swap                               swap (x,y);
  (var x: integer;                           writeln (x);
   var y : integer);                         writeln (y)
var z : integer;                             end.
begin
  z := x; x := y; y := z
end;
```

Problem 5

Show the output produced by executing the following Pascal program. Note that Pascal is statically (lexically) scoped.

```
program P;      procedure D;      begin
var n : char;  var n : char;      n := 'L';
procedure W;   begin              W;
begin          n := 'D';          D
  writeln(n)   W                  end.
end;          end;
```

Problem 6

Show the meaning of the following programs (1) and (2) by using the rules presented in the lecture. Note that the programs are in the small subset of C presented in the lecture. Let the states before executing the programs both to be $\sigma = \{(X, 3), (Y, 1), (Z, 0)\}$.

(1) 2

(2) ((2+3)*X)

Rules presented in the lecture

Hoare logic

$$\frac{\{P\} S_1 \{Q\} \quad \{Q\} S_2 \{R\}}{\{P\} S_1; S_2 \{R\}} \text{ (composition rule)}$$

$$\frac{\{P \wedge E\} S_1 \{Q\} \quad \{P \wedge \neg E\} S_2 \{Q\}}{\{P\} \text{ if } E \text{ then } S_1 \text{ else } S_2 \{Q\}} \text{ (conditional rule)}$$

$$\frac{\{P \wedge E\} S \{P\}}{\{P\} \text{ while } E \text{ do } S \{P \wedge \neg E\}} \text{ (while rule)}$$

$$\{Q[E/x]\} x := E \{Q\} \text{ (assignment axiom)}$$

$$\frac{P \Rightarrow P' \quad \{P'\} S \{Q'\} \quad Q' \Rightarrow Q}{\{P\} S \{Q\}} \text{ (consequence rule)}$$

Operational semantics for the small subset of C

- Rules for arithmetic expressions

- Sequences of numbers: $\langle n, \sigma \rangle \rightarrow m$ where m is an integer represented by the sequence of numbers n in the decimal representation.

- Variables: $\langle x, \sigma \rangle \rightarrow \sigma(x)$

- Addition:

$$\frac{\langle a_1, \sigma \rangle \rightarrow m_1 \quad \langle a_2, \sigma \rangle \rightarrow m_2}{\langle (a_1 + a_2), \sigma \rangle \rightarrow m} \text{ (} m \text{ is the sum of } m_1 \text{ and } m_2 \text{.)}$$

- Subtraction:

$$\frac{\langle a_1, \sigma \rangle \rightarrow m_1 \quad \langle a_2, \sigma \rangle \rightarrow m_2}{\langle (a_1 - a_2), \sigma \rangle \rightarrow m} \text{ (} m \text{ is the difference of } m_1 \text{ and } m_2 \text{.)}$$

- Multiplication:

$$\frac{\langle a_1, \sigma \rangle \rightarrow m_1 \quad \langle a_2, \sigma \rangle \rightarrow m_2}{\langle (a_1 * a_2), \sigma \rangle \rightarrow m} \text{ (} m \text{ is the product of } m_1 \text{ and } m_2 \text{.)}$$

- Rules for statements

- Assignments:

$$\frac{\langle a, \sigma \rangle \rightarrow m}{\langle x = a; \sigma \rangle \rightarrow \sigma[m/x]}$$

where $\sigma[m/x]$ is defined as follows.

$$(\sigma[m/x])(y) = \begin{cases} m & \text{if } y = x \\ \sigma(y) & \text{if } y \neq x \end{cases}$$

- Sequences:

$$\frac{\langle c_1, \sigma \rangle \rightarrow \sigma_1 \quad \langle c_2, \sigma_1 \rangle \rightarrow \sigma_2}{\langle c_1 \ c_2, \sigma \rangle \rightarrow \sigma_2}$$

- **while** statements:

$$\frac{\langle a, \sigma \rangle \rightarrow 0 \quad \langle \text{while } (a) \{c\}, \sigma \rangle \rightarrow \sigma}{\langle a, \sigma \rangle \rightarrow m \quad \langle c, \sigma \rangle \rightarrow \sigma_1 \quad \langle \text{while } (a) \{c\}, \sigma_1 \rangle \rightarrow \sigma_2 \text{ (if } m \neq 0 \text{)}}{\langle \text{while } (a) \{c\}, \sigma \rangle \rightarrow \sigma_2}$$