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.

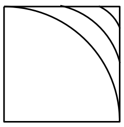

Figure 1: The quilt that `a` represents

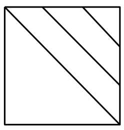

Figure 2: The quilt that `b` represents

- 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 (`e_1`, `e_2`)` represents the quilt that is obtained by sewing the two quilts e_1 and e_2, where e_1 is in the left side and e_2 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) $\{x = 5\}$ **while** $x > 0$ **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;
var x : integer;
var y : integer;
procedure swap
  (var x: integer;
   var y : integer);
var z : integer;
begin
   z := x; x := y; y := z
end;
```

```
begin
   x := 3;
   y := 4;
   swap (x,y);
   writeln (x);
   writeln (y)
end.
```

Problem 5

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

```
program P;
var n : char;
procedure W;
begin
   writeln(n)
end;
```

```
procedure D;
var n : char;
begin
   n := 'D';
   W
end;
```

```
begin
   n := 'L';
   W;
   D
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 $\sigma = \{(\mathtt{X}, 3), (\mathtt{Y}, 1), (\mathtt{Z}, 0)\}$.

(1) `Z=(X+4);`

(2) `while(Y){Y=(Y-1);}`

Rules presented in the lecture

Hoare logic

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

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

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

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

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

Operational semantics for the small subset of C

- Rules for arithmetic expressions
  - Sequences of numbers: $< n, \sigma > \rightarrow\ m$ where m is an integer represented by the sequence of numbers n in the decimal representation.
  - Variables: $< x, \sigma > \rightarrow\ \sigma(x)$
  - Addition:
  $$\frac{< a_1, \sigma > \rightarrow\ m_1 \quad < a_2, \sigma > \rightarrow\ m_2}{< (a_1 + a_2), \sigma > \rightarrow\ m} \text{ (m is the sum of m_1 and m_2.)}$$
  - Subtraction:
  $$\frac{< a_1, \sigma > \rightarrow\ m_1 \quad < a_2, \sigma > \rightarrow\ m_2}{< (a_1 - a_2), \sigma > \rightarrow\ m} \text{ (m is the difference of m_1 and m_2.)}$$
  - Multiplication:
  $$\frac{< a_1, \sigma > \rightarrow\ m_1 \quad < a_2, \sigma > \rightarrow\ m_2}{< (a_1 * a_2), \sigma > \rightarrow\ m} \text{ (m is the product of m_1 and m_2.)}$$
- Rules for statements
  - Assignments:
  $$\frac{< a, \sigma > \rightarrow\ m}{< x = a;, \sigma > \rightarrow\ \sigma[m/x]}$$
  where $\sigma[m/x]$ is defined as follows.
  $$(\sigma[m/x])(y) = \begin{cases} m & \textbf{if } y = x \\ \sigma(y) & \textbf{if } y \neq x \end{cases}$$
  - Sequences:
  $$\frac{< c_1, \sigma > \rightarrow\ \sigma_1 \quad < c_2, \sigma_1 > \rightarrow\ \sigma_2}{< c_1\ c_2, \sigma > \rightarrow\ \sigma_2}$$
  - **while** statements:
  $$\frac{< a, \sigma > \rightarrow\ 0}{< \textbf{while}\ (a)\ \{c\}, \sigma > \rightarrow\ \sigma}$$
  $$\frac{< a, \sigma > \rightarrow\ m \quad < c, \sigma > \rightarrow\ \sigma_1 \quad < \textbf{while}\ (a)\ \{c\}, \sigma_1 > \rightarrow\ \sigma_2}{< \textbf{while}\ (a)\ \{c\}, \sigma > \rightarrow\ \sigma_2} \text{ (\textbf{if} $m \neq 0$)}$$