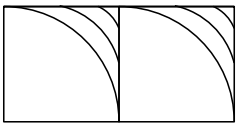
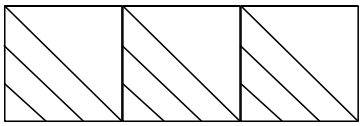
Foundations for Programming Languages
Answers for small examination 1

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

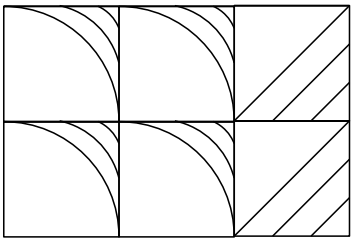
(1)
```
sew (a, a)
```



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



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



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

- Expressions `a` and `b` represent the quilts in Figure 1 and Figure 2 respectively.
- The expression `turn` (*e*) represents the quilt obtained by rotating 90 degrees to the right the quilt represented by the expression *e*.

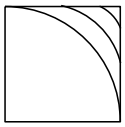


Figure 1: The quilt that `a` represents

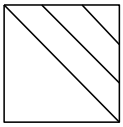


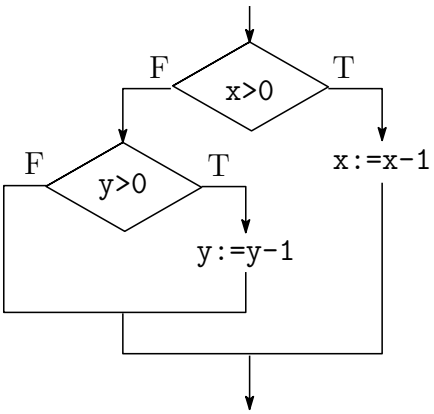
Figure 2: The quilt that `b` represents

- 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.

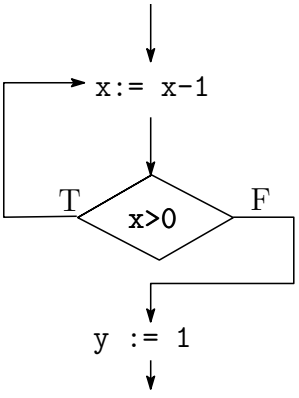
(1) Illustrate the control flow of the following program fragment.

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



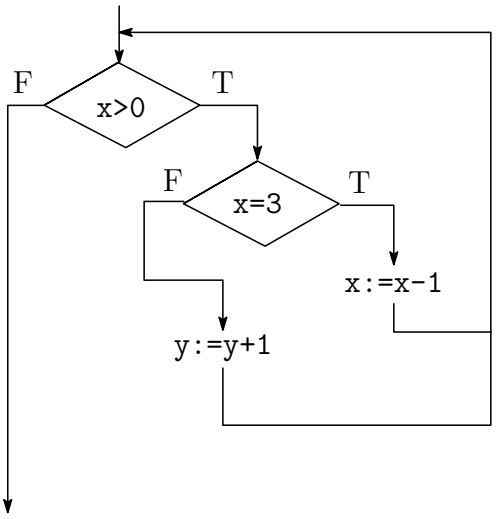
(2) Illustrate the control flow of the following program fragment.

```
L: x := x - 1;
if x>0 then goto L;
y := 1
```



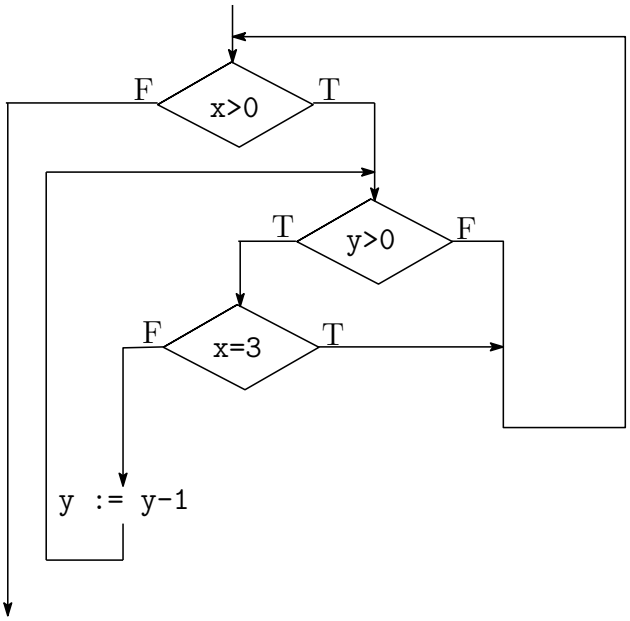
(3) Illustrate the control flow of the following program fragment.

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



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

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



(5) How many entries and exits does the if statement (`if x=3 then break;`) in the program fragment (4) have?

The if statement has one entry and two exits.

Problem 3

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

(1) $\{a = 40\}\ a := a - 20\ \{a = 20\}$

$$\dfrac{a = 40 \Rightarrow a - 20 = 20 \quad \dfrac{}{\{a - 20 = 20\}\ a := a - 20\ \{a = 20\}}\ (\text{assign}) \quad a = 20 \Rightarrow a = 20}{\{a = 40\}\ a := a - 20\ \{a = 20\}}\ (\text{conseq})$$

As I said in the lecture, the logical expression $a = 20 \Rightarrow a = 20$ in the above proof tree may be omitted in this class as follows.

$$\dfrac{a = 40 \Rightarrow a - 20 = 20 \quad \dfrac{}{\{a - 20 = 20\}\ a := a - 20\ \{a = 20\}}\ (\text{assign})}{\{a = 40\}\ a := a - 20\ \{a = 20\}}\ (\text{conseq})$$

(2) $\{a = 3\}\ a := a + 1; a := a + 3\ \{a = 7\}$

$$\dfrac{\dfrac{a = 3 \Rightarrow a + 1 = 4 \quad \dfrac{}{\{a + 1 = 4\}\ a := a + 1\ \{a = 4\}}\ (\text{assign})}{\{a = 3\}\ a := a + 1\ \{a = 4\}}\ (\text{conseq}) \quad \dfrac{a = 4 \Rightarrow a + 3 = 7 \quad \dfrac{}{\{a + 3 = 7\}\ a := a + 3\ \{a = 7\}}\ (\text{assign})}{\{a = 4\}\ a := a + 3\ \{a = 7\}}\ (\text{conseq})}{\{a = 3\}\ a := a + 1; a := a + 3\ \{a = 7\}}\ (\text{composition})$$

In this proof, I omitted $a = 4 \Rightarrow a = 4$ and $a = 7 \Rightarrow a = 7$ in the applications of the consequence rule.

(3) $\{a = 4\}$ **if** $a = 4$ **then** $a := a + 3$ **else** $a := a - 3\ \{a = 7\}$

Due to space restriction, I write the proof tree by separating it into three parts.

$$\dfrac{\dfrac{\text{(I write this part below.)}}{\{a=4 \wedge a=4\}\ a := a+3\ \{a=7\}}\ \text{(conseq)} \quad \dfrac{\text{(I write this part below.)}}{\{a=4 \wedge \neg\ a=4\}\ a := a-3\ \{a=7\}}\ \text{(conseq)}}{\{a=4\}\ \textbf{if}\ a=4\ \textbf{then}\ a := a+3\ \textbf{else}\ a := a-3\ \{a=7\}}\ \text{(conditional)}$$

$$\dfrac{a=4 \wedge a=4 \Rightarrow a+3=7 \quad \dfrac{}{\{a+3=7\}\ a := a+3\ \{a=7\}}\ \text{(assign)}}{\{a=4 \wedge a=4\}\ a := a+3\ \{a=7\}}\ \text{(conseq)}$$

$$\dfrac{a=4 \wedge \neg\ a=4 \Rightarrow a-3=7 \quad \dfrac{}{\{a-3=7\}\ a := a-3\ \{a=7\}}\ \text{(assign)}}{\{a=4 \wedge \neg\ a=4\}\ a := a-3\ \{a=7\}}\ \text{(conseq)}$$

(4) $\{a=2\}$ **while** $a<5$ **do** $a := a+1\ \{a=5\}$

Due to space restriction, I write the proof tree by separating it into two parts.

$$\dfrac{a=2 \Rightarrow a \leq 5 \quad \dfrac{\text{(I write this part below.)}}{\{a \leq 5\}\ \textbf{while}\ a<5\ \textbf{do}\ a := a+1\ \{a \leq 5 \wedge \neg\ a<5\}} \quad a \leq 5 \wedge \neg\ a<5 \Rightarrow a=5}{\{a=2\}\ \textbf{while}\ a<5\ \textbf{do}\ a := a+1\ \{a=5\}}\ \text{(conseq)}$$

$$\dfrac{\dfrac{a \leq 5 \wedge a<5 \Rightarrow a+1 \leq 5 \quad \dfrac{}{\{a+1 \leq 5\}\ a := a+1\{a \leq 5\}}\ \text{(assign)} \quad a \leq 5 \Rightarrow a \leq 5}{\{a \leq 5 \wedge a<5\}\ a := a+1\{a \leq 5\}}\ \text{(conseq)}}{\{a \leq 5\}\ \textbf{while}\ a<5\ \textbf{do}\ a := a+1\ \{a \leq 5 \wedge \neg\ a<5\}}\ \text{(while)}$$

In the above proof tree, the logical expression $a \leq 5 \Rightarrow a \leq 5$ may be omitted as follows.

$$\dfrac{\dfrac{a \leq 5 \wedge a<5 \Rightarrow a+1 \leq 5 \quad \dfrac{}{\{a+1 \leq 5\}\ a := a+1\{a \leq 5\}}\ \text{(assign)}}{\{a \leq 5 \wedge a<5\}\ a := a+1\{a \leq 5\}}\ \text{(conseq)}}{\{a \leq 5\}\ \textbf{while}\ a<5\ \textbf{do}\ a := a+1\ \{a \leq 5 \wedge \neg\ a<5\}}\ \text{(while)}$$

I abbreviated the assignment axiom as assign, the consequence rule as conseq, the while rule as while, and the composition rule as composition.