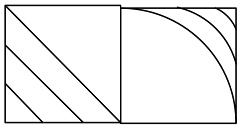
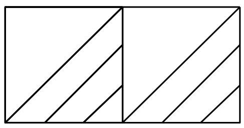
Principles of Programming Languages
Answers for small examination 1

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

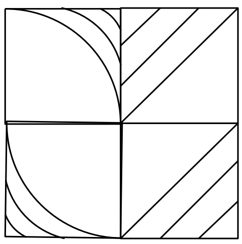
(1)
```
sew (turn (turn (b)), a)
```



(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` 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.
- 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.

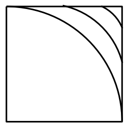


Figure 1: The quilt that `a` represents

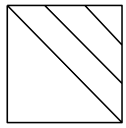
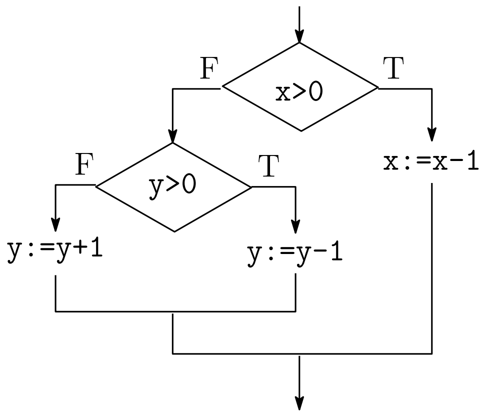


Figure 2: The quilt that `b` represents

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

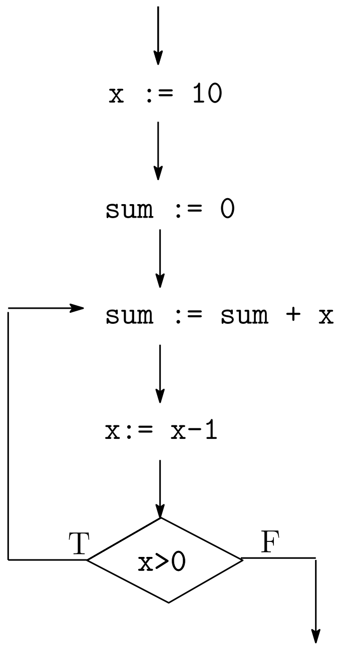
(1) Illustrate the control flow of the following program fragment.

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



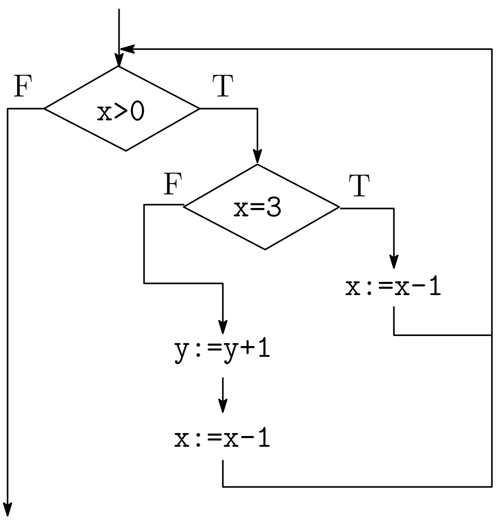
(2) Illustrate the control flow of the following program fragment.

```
x := 10;
sum := 0;
L:  sum := sum + x;
x := x - 1;
if x>0 then
    goto L
```



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

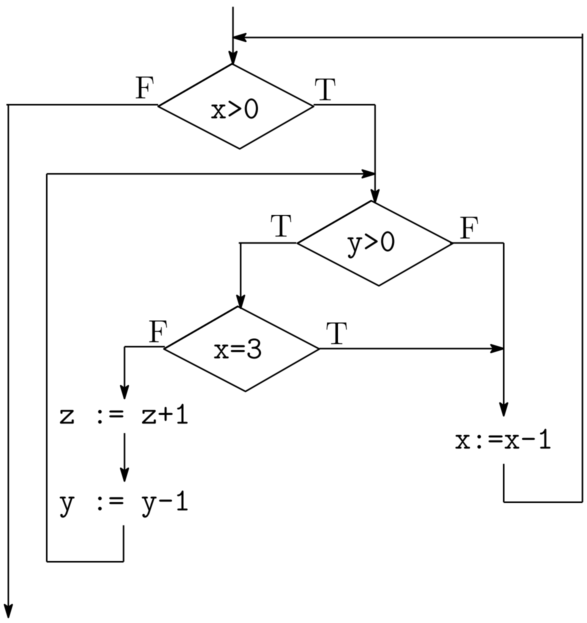
```
while x>0 do
    begin
        if x=3 then
            begin
                x := x - 1;
                continue
            end;
        y := y + 1;
        x := x - 1
    end
```



(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;
                z := z + 1;
                y := y - 1
            end;
        x := x - 1
    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) $\{a = 3\}\ a := a + 1\ \{a = 4\}$

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

As I said in the lecture, the logical expression $a = 4 \Rightarrow a = 4$ in the above proof tree may be omitted in this class as follows.

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

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

$$\dfrac{\dfrac{a=3\Rightarrow a+1=4 \quad \dfrac{}{\{a+1=4\}\ a:=a+1\ \{a=4\}}\text{(assign)}}{\{a=3\}\ a:=a+1\ \{a=4\}}\text{(conseq)} \quad \dfrac{a=4\Rightarrow a+2=6 \quad \dfrac{}{\{a+2=6\}\ a:=a+2\ \{a=6\}}\text{(assign)}}{\{a=4\}\ a:=a+2\ \{a=6\}}\text{(conseq)}}{\{a=3\}\ a:=a+1; a:=a+2\ \{a=6\}}\text{(composition)}$$

In this proof, I omitted $a=4 \Rightarrow a=4$ and $a=6 \Rightarrow a=6$ in the applications of the consequence rule.

(3) $\{a=4\}$ **if** $a=4$ **then** $a:=a+2$ **else** $a:=a-3\ \{a=6\}$

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

$$\dfrac{\dfrac{\text{(I write this part below.)}}{\{a=4\wedge a=4\}\ a:=a+2\ \{a=6\}}\text{(conseq)} \quad \dfrac{\text{(I write this part below.)}}{\{a=4\wedge\neg\, a=4\}\ a:=a-3\ \{a=6\}}\text{(conseq)}}{\{a=4\}\ \textbf{if}\ a=4\ \textbf{then}\ a:=a+2\ \textbf{else}\ a:=a-3\ \{a=6\}}\text{(conditional)}$$

$$\dfrac{a=4\wedge a=4\Rightarrow a+2=6 \quad \dfrac{}{\{a+2=6\}\ a:=a+2\ \{a=6\}}\text{(assign)}}{\{a=4\wedge a=4\}\ a:=a+2\ \{a=6\}}\text{(conseq)}$$

$$\dfrac{a=4\wedge\neg\, a=4\Rightarrow a-3=6 \quad \dfrac{}{\{a-3=6\}\ a:=a-3\ \{a=6\}}\text{(assign)}}{\{a=4\wedge\neg\, a=4\}\ a:=a-3\ \{a=6\}}\text{(conseq)}$$

(4) $\{a=5\}$ **while** $a>0$ **do** $a:=a-1\ \{a=0\}$

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

$$\dfrac{a=5\Rightarrow a\geq 0 \quad \dfrac{\text{(I write this part below.)}}{\{a\geq 0\}\ \textbf{while}\ a>0\ \textbf{do}\ a:=a-1\ \{a\geq 0\wedge\neg\, a>0\}} \quad a\geq 0\wedge\neg\, a>0\Rightarrow a=0}{\{a=5\}\ \textbf{while}\ a>0\ \textbf{do}\ a:=a-1\ \{a=0\}}\text{(conseq)}$$

$$\dfrac{\dfrac{a\geq 0\wedge a>0\Rightarrow a-1\geq 0 \quad \dfrac{}{\{a-1\geq 0\}\ a:=a-1\{a\geq 0\}}\text{(assign)} \quad a\geq 0\Rightarrow a\geq 0}{\{a\geq 0\wedge a>0\}\ a:=a-1\{a\geq 0\}}\text{(conseq)}}{\{a\geq 0\}\ \textbf{while}\ a>0\ \textbf{do}\ a:=a-1\ \{a\geq 0\wedge\neg\, a>0\}}\text{(while)}$$

In the above proof tree, the logical expression $a\geq 0 \Rightarrow a\geq 0$ may be omitted as follows.

$$\dfrac{\dfrac{a\geq 0\wedge a>0\Rightarrow a-1\geq 0 \quad \dfrac{}{\{a-1\geq 0\}\ a:=a-1\{a\geq 0\}}\text{(assign)}}{\{a\geq 0\wedge a>0\}\ a:=a-1\{a\geq 0\}}\text{(conseq)}}{\{a\geq 0\}\ \textbf{while}\ a>0\ \textbf{do}\ a:=a-1\ \{a\geq 0\wedge\neg\, a>0\}}\text{(while)}$$

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