**Lexical analysis**

Describe a lexical analyzer that recognizes a sequence of the following tokens using the longest match rule: $<=$, $=>$, $<=>$, $=$, $==$, $<=$.  
It should also skip whitespace characters between the tokens. (Denote the whitespace characters by the \_ (underscore) symbol.)  
As an example, consider the string $<==><==<=>$. It should give the token stream $<=$, $=>$, $<=$, $=$, $<=>$

* Draw an automaton which accepts the tokens above, optionally followed by white spaces. Use one diff  
  erent accepting state for each token.

**Parsing**

Let’s consider a grammar for expressions where we want to make the multiplication sign optional.

ex ::= ex + ex | ex \* ex | ex \_ ex | ex / ex | ( ex ) | ID | INTLITERAL

The underscore denotes whitespace.

* Find a LL(1) grammar recognizing the same language.
* Using your grammar, draw the parse tree for the following expression: 3 \* x - z \_ 2
* Create the LL(1) parsing table.

**Type checking**

If the following programs type-check according to the rules given in the course, give the corresponding type derivation tree, otherwise give a partial tree that shows where it doesn’t work

class A {}

class B extends A {}

class Test {

val array: Array[A] = new Array[B](2)

array(0) = new A

array(0) = new B

}

class Test {

class A {}

class B extends A {}

class C extends B {}

def func(x: B=>B): A

val y: A=>C

val z = func(y)

}

**Code generation**

Translate the following code into bytecode using the techniques seen in class:

boolean b;

int f(int x, int y, int z) {

while ((!b && (x > 2\*(y+z))) || (x < 2\*y +z)) {

x = x +3;

}

return x;

}

**Data-flow analysis**

Draw the control-flow graph for the following program.

var x = 2

var y = input()

if (x == y) {

do {

y = y + 1

x = x + y + 3

} while (y < 4)

} else if (y <= -1 && y >= -7) {

y = y \* y

} else {

y = x - 3

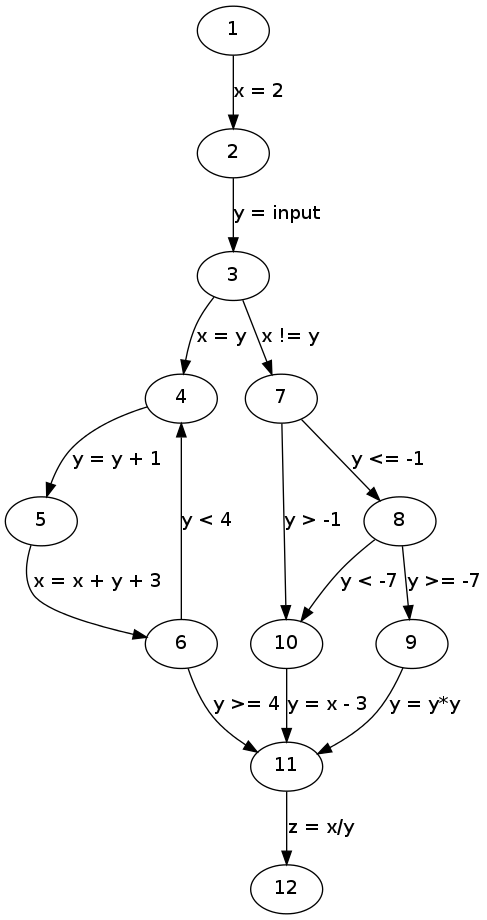
}

val z = x/y

Assume that the ranges that the integers can take are [-128, 127] for simplicity, i.e. all intervals within this range are allowed, for instance [-1, 3]. The function input() every time returns a possibly different integer in [-128, 127], and this integer is not known at compile time.

Run range analysis that maintains an interval for x and y at each program point in the control flow graph. Give the intervals for each node in the control-flow graph obtained after the analysis.

What are the values of all variables at the last assignment?

Solution: [](http://lara.epfl.ch/w/_detail/cc12:cfg.png?id=cc12%3Aquiz_preparation&cache=cache)

and the final ranges are:

x y

1: bottom bottom

2: [2, 2] [-128, 127]

3: [2, 2] [-128, 127]

4: [2, 127] [2, 3]

5: [2, 127] [3, 4]

6: [8, 127] [3, 4]

7: [2, 2] [-128, 127]

8: [2, 2] [-128, -1]

9: [2, 2] [-7, -1]

10: [2, 2] [-128, 127]

11: [2, 127] [-1, 49]

12: [2, 127] [-1, 49] z: T

Thus, in the final assignment, y can have value 0 and a division by zero is possible (according to the analysis).