COMPILERS

Final Project

May 2021

1 Add comparison operators to the infix calculator

The infix calculator introduced in class needs to be enriched with comparisons between operands. The new operators to be added are:

$$<, >, <=, >=, ==, != \tag{1}$$

Which, in order, represent "less than", "more than", "less than or equal to", "more than or equal to", "equal to" and "different from". For these operators to work you will need to introduce two new values to the language:

$$True, False \tag{2}$$

With the new operands and values in our language you will be able to write code like this:

```
a = 2
b = 3
a <= b
True
a > (b - 1)
False
a == b
False
```

This exercise is worth: **1 point**.

2 Add if-then-else statements to the infix calculator

Using the boolean values and comparison operators from the previous exercise, you need to add **if** statements to our infix calculator with this notation:

```
if BOOLEAN_CONDITION then OP1 else OP2
```

This will allow you to run code like this:

```
a = 2
b = 4
if (b < a) then a - b else b - a
2
```

And also:

```
if False do 9 - 3 else 9 + 3
12
```

Notice that our boolean condition can either be a comparison operation between numbers (or variables) or a boolean value.

This exercise is worth: **1 point**.

3 Translate the infix calculator notation into 3-address code

For this exercise, the compiler will need to accept as input a file containing code written with the syntax of our new infix calculator (with comparison operations and if-then-else statements). The output of the compiler will be a translation of the infix calculator notation into a 3-address code written with C syntax (bonus points if it compiles with gcc).

For example, assume that this code is given in input to our new compiler:

```
a = 2
b = 3
c = 8
d = c - (a * b)
if d == 2 then a + b + c else a - b - c
```

Then the compiler would output the following program, written in 3-address code with C syntax:

```
void main(){
int a, b, c, t1, t2, t3, t4;

a = 2;
b = 3;
c = 8;
t1 = a * b;
t2 = c - t1;

if(t2 == 2)
  goto first;
else
  goto second;

first:
t3 = a + b;
t4 = t3 + c;
printf("%d\n", t4);

Second:
t3 = b - c;
t4 = a - t3;
printf("%d\n", t4);
}
```

This exercise is worth: **2 points**.