Chair of Applied Computer Science III
University of Mannheim

Prof. Dr. Guido Moerkotte
Email: moerkotte@uni-mannheim.de

Daniel Flachs
Email: daniel.flachs@uni-mannheim.de

Database Systems II — Solution to Exercise Sheet 8
Spring Semester 2019 — Created May 7, 2019

Exercise 1

Building an operator tree is one possibility to interpret a program. Every operator is encapsulated in a class and all operators share a common interface. In our case, this is the `eval` function. In the files provided on the website, you can find a simple implementation of operators to build an operator tree for expressions in Boolean algebra. You can also find a small example. So far, the following operations exist:

- `Literal` which just creates a literal like `a := true`,
- `Not`, which negates an expression, e.g. `Not(a)`,
- `Or`, the logical or operator, to evaluate disjunctions,
- `Implication`, which evaluates an implication, like $a \Rightarrow b$.

Exercise 1 a)

Implement the logical `And` operator.

Exercise 1 b)

Construct and evaluate the following expression:

$$(a \wedge b) \vee c$$

You can choose values for $a, b, c \in \{\text{true}, \text{false}\}$.

Solution

See code.

Exercise 2

Another possibility to interprete a program is to run it in a virtual machine. In the files provided on the website, you can find a simple implementation of a virtual machine that supports only integer operations. The virtual machine is inspired by the IJVM (`https://en.wikipedia.org/wiki/IJVM`).

Exercise 2 a)

Look at the implementation.

Exercise 2 b)

The following example program is given in the code.

```
PUSH, 111,
DUPLICATETOP,
PRINTTOP,           // JMP jumps here
DUPLICATETOP,
PUSH, 122,
SUB,
JMPIFFALSE, 16,
PUSH, 1,
ADD,
JMP, 3,
POP,                // JMPIFFALSE jumps here
HALT
```

Give C++ code that is equivalent to the above series of instructions.

Solution

```
for (int i = 111; i <= 122; ++i) {
    std::cout << i << std::endl;
}
```

Exercise 2 c)

(i) Implement a multiplication operation `MUL` that takes the last two elements from the stack, removes them, and pushes their product onto the stack.

(ii) Implement an integer division operation `DIV`, that takes the last two elements from the stack, removes them, and pushes their quotient onto the stack. Make sure that the order of the operands is consistent with how the SUB operations handles its operands.

(iii) Also implement a conditional jump operation `JMPIFTRUE` that removes the last element from the stack and jumps to a specified location in the program if the element evaluates to `true`, i. e., a number unequal to zero.

Solution

See code.

Exercise 2 d)

Replace `program` with a program that computes

$$(5 + 3 \cdot 4) \div 2$$

and outputs the result.

Solution

```
int program [] = {
   PUSH, 2,
   PUSH, 5,
   PUSH, 3,
   PUSH, 4,
   MUL,
   ADD,
   DIV,
   PRINTTOP,
   POP,
   HALT
};
```