Assignment 8: Abstract machines for Call-by-value and Call-by-name

Consider a tiny language consisting of expressions that are

e ::= x | \x.e | e_1 (e_2)

{

In addition, to make things interesting, you can add the booleans T and F, and an if_then_else expression.

You can also introduce the natural numbers, addition (perhaps multiplication) and comparison with 0.

}

For the language you take, you should design and implement (in OCaml) the following abstract machines

1. The Krivine Machine (in closure form), that implements Call-by-Name semantics.

For this you need to consider Closures as a pair of a Table and an expression, where a Table is a partial function from variables to Answers (including value closures).

2. The SECD machine that implements Call-by-Value semantics.

For this you need value closures in the set of answers.

You also need to implement the compile function.

Most importantly, you need to provide inputs that demonstrate that your implementations of the two machines run correctly.

In case you implement recursion, for the extended languages you should be able to run factorial or fibonacci functions.