4 Compiler Construction (TGG)

Consider writing a compiler for a simple language of expressions given by the following grammar,

\[ e ::= n \quad (\text{integer}) \]
\[ \quad | \quad ? \quad (\text{read integer input from user}) \]
\[ \quad | \quad e + e \quad (\text{addition}) \]
\[ \quad | \quad e - e \quad (\text{subtraction}) \]
\[ \quad | \quad e \times e \quad (\text{multiplication}) \]
\[ \quad | \quad (e, e) \quad (\text{pair}) \]
\[ \quad | \quad \text{fst} \ e \quad (\text{first projection}) \]
\[ \quad | \quad \text{snd} \ e \quad (\text{second projection}) \]

(a) Describe the tasks that should be carried in implementing a front end for this language and any difficulties that might be encountered. [5 marks]

(b) Suppose that the target virtual machine is stack-oriented and that the stack elements are integer values, and addresses can be stored as integers. Explain which other features are required in such a virtual machine. Invent a simple language of instructions for such a machine and show how it would be used to implement each of the expressions. [10 marks]

(c) Suppose that the following rules are proposed as possible optimizations to be implemented in your compiler.

<table>
<thead>
<tr>
<th>expression simplifies to expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\text{fst} \ e, \text{snd} \ e)) \rightarrow e</td>
</tr>
<tr>
<td>\text{fst} \ (e_1, e_2) \rightarrow e_1</td>
</tr>
<tr>
<td>\text{snd} \ (e_1, e_2) \rightarrow e_2</td>
</tr>
</tbody>
</table>

Describe how you could implement these rules so that the simplifications are made only when the program’s semantics is correctly preserved. [5 marks]