4 Compiler Construction (TGG)

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

\[
\begin{align*}
e & ::= n \quad \text{(integer)} \\
& \mid ? \quad \text{(read integer input from user)} \\
& \mid e + e \quad \text{(addition)} \\
& \mid e - e \quad \text{(subtraction)} \\
& \mid e \ast e \quad \text{(multiplication)} \\
& \mid (e, e) \quad \text{(pair)} \\
& \mid \text{fst } e \quad \text{(first projection)} \\
& \mid \text{snd } e \quad \text{(second projection)}
\end{align*}
\]

(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.

\[
\begin{align*}
\text{expression} & \quad \text{simplifies to} \quad \text{expression} \\
\text{fst } e, \text{ snd } e & \rightarrow e \\
\text{fst } (e_1, e_2) & \rightarrow e_1 \\
\text{snd } (e_1, e_2) & \rightarrow e_2
\end{align*}
\]

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