4 Compiler Construction (TGG)

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

$$
\begin{array}{rcll}
e & ::= & n & \text{(integer)} \\
 & | & ? & \text{(read integer input from user)} \\
 & | & e + e & \text{(addition)} \\
 & | & e - e & \text{(subtraction)} \\
 & | & e * e & \text{(multiplication)} \\
 & | & (e, e) & \text{(pair)} \\
 & | & \texttt{fst}\ e & \text{(first projection)} \\
 & | & \texttt{snd}\ e & \text{(second projection)}
\end{array}
$$

(*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.

expression	simplifies to	expression
$(\texttt{fst}\ e,\ \texttt{snd}\ e)$	$\rightarrow$	e
$\texttt{fst}\ (e_1,\ e_2)$	$\rightarrow$	e_1
$\texttt{snd}\ (e_1,\ e_2)$	$\rightarrow$	e_2

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