Compiler Construction

Explain how a parse-tree representation of a program may be converted into a stack-based intermediate language giving sketches of code to translate expressions, assignments and the if-then-else command; you should also explain how occurrences of a variable in an expression or assignment are translated.

The program may be assumed to conform to the following syntax:

\[
\begin{align*}
E & \rightarrow \ n \mid x \mid E + E \mid f(E,E) \\
D & \rightarrow \ let \ f(x,x) = \{Dseq; Cseq; E\} \mid let \ x = E \\
C & \rightarrow \ x := E; \mid if \ E \ then \ C \ else \ C \\
Cseq & \rightarrow \ C \mid C \ Cseq \\
Dseq & \rightarrow \ D \mid D \ Dseq
\end{align*}
\]

with start symbol \(Dseq\). Here \(n\) corresponds to integer constants, \(x\) corresponds to identifiers used as variable names and \(f\) corresponds to identifiers used as function names (you may assume these are disjoint). The function declaration construct has the effect of defining a function which, when called, makes declarations, performs commands and then returns the result of its expression; note that therefore functions may be defined within functions, but the above restriction on identifiers means that they cannot be returned as results.  

[20 marks]