1. Given a compiler for language $S$, which runs on machine $M$ and produces Intermediate code $I$. You have also another compiler for $I$ that is running on machine $M$ and produces machine code of $M$. Show how to generate another compiler for $S$ to run on machine $M$ and produce code for $M$. (5 marks)

2.a) Write a regular expression for an identifier that starts with a letter and then followed by letter, digit, or underscore _ followed by letter or digit. Examples for correct identifiers, $a, a345, a45_{y}, asd_{t}, as5_{t7}$ (3 marks)

b) convert the regular expression into DFA (2 marks)
3. Given the following grammar into EBNF notation:
   <for_stmt> --> for <var> := <list_element> [, <list_element>] do <statement>
   <list_element> --> <expr> [step <expr> until <expr>]

   a) Rewrite the grammar into BNF notation (3 marks)

   b) Assume that the <var> → id, <expr> → e1 | e2 | e3 | e4, and <statement> → s, draw the parse tree for the statement

      for id:= e1 step e2 until e2, e4 do s (2 marks)
4. Consider the following grammar:

\[
\begin{align*}
\text{D} & \rightarrow \text{var} \ V : \ T \\
\text{V} & \rightarrow \text{id}, \ V | \text{id} \\
\text{T} & \rightarrow \text{integer} | \text{real}
\end{align*}
\]

a) Left factor this grammar (2 marks)

b) Construct first and follow sets for non-terminals of the resulting grammar (4 marks)

c) Construct the LL(1) parsing table for the resulting grammar (2 marks)
d) Show the actions of the corresponding LL(1) parser, given the input string
\textbf{var} \textbf{x,y : integer} (i.e. trace the parser showing the parsing stack and the processed input) (2 marks)
5. Write a parser for the above grammar, after left factoring, using recursive decent method. (5 marks)