1. Given a compiler for language S, which runs on machine M and produces code for M. Show how to generate another compiler for S to run on machine N and produce code for N. The generation process must be done on machine A. 

(5 marks)

2. Write down the DFA for the numeric constants in scientific notation given the following regular definition:

\[
\begin{align*}
\text{digit} & = [0-7] \\
\text{nat} & = \text{digit} + \\
\text{signedNat} & = (+|-) \ \text{nat} \\
\text{number} & = \text{signedNat} (\cdot \ \text{nat})?\text{E} \ \text{signedNat})?
\end{align*}
\]

(5 marks)
3. Show that the following grammar is ambiguous

\[
S \rightarrow \text{if } (E) S | M \\
M \rightarrow \text{if } (E) S \text{ else } M | \text{other} \\
E \rightarrow \text{id } R \text{id} \\
R \rightarrow >|<|= 
\]

(5 marks)
4. Consider the following grammar:

\[
\text{Sexp} \rightarrow \text{id} \mid (\text{Sexp-list op}) \\
\text{op} \rightarrow + \mid - \mid * \\
\text{Sexp-list} \rightarrow \text{Sexp-list Sexp} \mid \text{Sexp}
\]

a) Remove the left recursion

(1 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

(4 marks)
d) Show the actions of the corresponding LL(1) parser, given the input string

\[(\text{id id id }-)^*\]

(2 marks)

e) Write pseudo code to parse the above grammar using recursive-descent

(4 marks)