1. What are the advantages of compiled models of a high level programming language over interpreted models? [4 marks]

2. What are the advantages of generating machine code over generating intermediate code? [4 marks]

3. Show the steps to develop a compiler for language L to produce code for machine A and run on machine A using bootstrapping technique. [4 marks]
4. Describe the phases of program translation from source code to machine code and the files produced by these phases and their contents in case of multi-pass compiler. [6 marks]

5. What are the advantage and disadvantage of multi-pass compiler over single-pass compiler? [2 marks]
6. **Given the following informal definition for a floating number in scientific notation:**
   - The mantissa may have a decimal point, and
   - The exponent may have a sign.
     
     Examples of correct numbers are: 3E6, 3.8E-6
     Examples of incorrect numbers: 3E, E+5, 2E3.4

   a. Write a regular expression for floating number in scientific notation.
      
      [5 marks]

   b. Convert the regular expression into DFA
      
      [5 marks]

   c. Write a pseudo code that implements the DFA constructed in (b)
      
      [5 marks]
7. Write a grammar for Boolean expressions that includes the operators “and”, and “or”. Be sure to give “or” a higher precedence than “and”. The Boolean variable is identifier or a Boolean constant (true or false). Parenthesis can give the operator of any sub expression higher priority. Also be sure your grammar is not ambiguous. [4 marks]

8. Consider the following grammar representing simplified LISP-like expressions:

```plaintext
lexp --> atom | list
atom --> number | identifier
list --> ( lexp-seq)
lexp-seq --> lexp-seq lexp | lexp
```

Show the first five steps of the leftmost and rightmost derivations for the string 

(a 23 (m x y)). [6 marks]
9. Consider the following grammar:

\[\begin{align*}
L & \rightarrow L \mid S \\
S & \rightarrow \text{id} \mid \text{id}\ E \mid \text{id}\ (E) \\
E & \rightarrow \text{id} \mid \text{num}
\end{align*}\]

a) Remove left recursion and left factor the grammar to be LL(1) [4 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
\texttt{id = num}  (i.e. trace the parser showing the parsing stack and the processed input)

[3 marks]