1 Lexical Analysis

This problem includes 2 questions for a total of $5 + 5 = 10$ points.

Consider this NFA:

![NFA Diagram]

Question 1.1 (5 points). Show a DFA that accepts the same language as the NFA (1). You can show your DFA either graphically, or by providing a transition table, whichever you prefer. Explain why the DFA defines the same language as the NFA.

Question 1.2 (5 points). Show a regular expression that accepts the same language as the NFA (1). Explain why the regular expression defines the same language as the NFA.

2 Syntax Analysis

This problem includes 3 questions for a total of $3 + 2 + 5 = 10$ points.

The programming language Fake1 has the following expression grammar:

- $E \rightarrow E < S | S$
  (comparison)
- $S \rightarrow S + P | P$
  (sum)
- $P \rightarrow \text{num} | (E)$
  (primary)

where \text{num} stands for usual integer number tokens.

Question 2.1 (3 points). Make a picture of the parse tree obtained from the Fake1 grammar by parsing the expression $1 < 2 < 3 + 4$.

Question 2.2 (2 points). Explain why the Fake1 grammar is left recursive.

Question 2.3 (5 points). Write a new grammar for Fake1 where left recursion is eliminated.
3 Name Analysis

This problem includes 3 questions for a total of 5+3+2 = 10 points.

Consider the following Tiger program:

```tiger
let
var v := "global"
function foo() = print(v)
function bar() = let var v := "local" in foo() end
in
bar()
end
```

**Question 3.1** (5 points). What is the symbol table for this program, i.e., the complete tree of scopes? For each scope, list the symbols defined in that scope!

**Question 3.2** (3 points). What does the program print? Briefly explain!

**Question 3.3** (2 points). What would the program print with dynamic scoping? Briefly explain!

4 Type Analysis

This problem includes 2 questions for a total of 2+8 = 10 points.

Consider again the programming language Fake1 with the expression grammar

\[
E \rightarrow E < S \mid S \\
S \rightarrow S + P \mid P \\
P \rightarrow \text{num} \mid (E)
\]

where num stands for usual integer number tokens.

Fake1 is slightly unusual in that it permits “chained” inequalities like \(1 < 2 < 3\), which means that the two inequalities \(1 < 2\) and \(2 < 3\) both hold. The grammar also allows undefined expressions like \((1 < 2) + (3 < 2)\).

**Question 4.1** (2 points). Explain the purpose of having a type system for a language such as Fake1.

**Question 4.2** (8 points). Define a syntax directed translation (SDT) that associates a type attribute to each Fake1 expression parse tree node; the special type value \text{TypeError} should be used for ill-typed expressions, which include \((1 < 2) + (3 < 2)\) as well as \((1 < 2) < 3\). (Note: it is alright to use “if-then-else” in the semantic rules.) Explain any special difficulties you have to solve.

This is the end of this exam.