Questions

6 questions, total 60 points.

1. Lexical analysis (10 points).
   Consider the following NFA:

   ![NFA Diagram]

   Show a DFA that accepts the same language as the NFA.

2. Syntax analysis (10 points).
   Let grammar G1 be defined as:
   
   \[
   T \rightarrow T + F | F \\
   F \rightarrow 1 | 2 | 3 
   \]

   Let grammar G2 be defined as:
   
   \[
   T \rightarrow F R \\
   R \rightarrow + F R | \epsilon \\
   F \rightarrow 1 | 2 | 3 
   \]

   2a. (2 points). Draw the parse tree for the string 1+2+3 using grammar G1.
   2b. (2 points). Draw the parse tree for the string 1+2+3 using grammar G2.
   2c. (6 points). Give an SDT scheme that converts trees parsed by grammar G2 into trees for grammar G1. You can use the function \( \text{newT(t}_l, \text{op}, t_r) \) to create a new AST node for non-terminal T, using left subtree \( t_l \), operator \( \text{op} \), and right subtree \( t_r \).

3. Type analysis (10 points).
   Consider the following types in the Tiger language.

   \[
   \begin{align*}
   \text{type A} &= \{ \text{i : int, p : A} \} \\
   \text{type B} &= \{ \text{i : int, q : B} \} \\
   \text{type C} &= \{ \text{p : C, is : array of int} \} \\
   \text{type D} &= \{ \text{p : D, is : array of int} \} \\
   \text{type E} &= \{ \text{p : E, q : F} \} \\
   \text{type F} &= \{ \text{p : F, q : E} \}
   \end{align*}
   \]

   3a. (3 points). Are types A and B structurally equivalent?
   3b. (3 points). Are types C and D structurally equivalent?
   3c. (4 points). Are types E and F structurally equivalent?

4. IR generation (10 points).
   Consider the following program fragment, which is written in Tiger extended with an additional feature, the \texttt{do-until} loop:

   ![Program Fragment]
e := 1;
i := 3;
do (
    e := e + e;
i := i - 1
) until i = 0;
e := e - 1;

Informally, the do-until loop first executes its body. Then it checks the condition. If the condition is true, it exits; otherwise, it repeats.

4a. (2 points). What is the final value in variable e at the end?
4b. (8 points). Translate the program fragment to Tiger IR.

5. Runtime environment (10 points).
Consider the following picture of an activation record, which illustrates the same x64/Linux calling conventions we have been discussing in class:

5a. (2 points) How does the return address get on the stack?
5b. (2 points) Where does the callee’s %rbp point?
5c. (2 points) How does the “caller %rbp” get on the stack?
5d. (2 points) Which slots are written by the caller?
5e. (2 points) Which slots are written by the callee?

6. Register allocation (10 points).
Consider the following control flow graph with Tiger IR code:

6a. (7 points) Draw the interference graph for the variables in this code.
6b. (3 points) Briefly explain how the interference graph gets used in register allocation.

This is the end of the exam.