Time 60min. Open notes/books. Use extra paper as needed, but make sure to identify each answer. All questions equal. **If something is not clear, state a reasonable assumption and then answer the question.**

**YOU MUST RETURN THIS PAGE. NAME___________________________**

1. (Use diagram notation) You have a compiler for C which runs slow but generates super efficient optimized code. You write a program PrgC in C.
   a) Show how you accomplish generative execution?
   b) What is the slow process in a)
   c) How can you improve that slow process assuming you have access to source code for your compiler?

2. (Use diagram notation) Suppose you developed a new language D. The only compiler you have is a standard 2 stage Pascal compiler+interpreter. You write your first program PrgD in D. You want to execute it.
   a) What is the fastest way to get it executed (not the fastest execution but the fastest way to get to execute the program)?
   b) Is the above an efficient way for the long run? If not, how would you improve it so that people want to use your language?

3. Suppose we deal with ternary alphabet {0,1,2}. Tokens come in separated by WS. The language is that of strings such that no input digit can be bigger than any previous digit in the same string. For example, **00122** or **111** are valid strings but **102** is not. Design a DFA to determine if an input string is a valid string.

4. Design an unambiguous grammar for boolean operators **AND, OR** and **NOT**. AND and OR are binary, NOT is unary. We want to have the standard precedence as in C or logic (NOT, AND, OR, strongest to weakest). However, AND is to associate left to right, OR the opposite way. NOT is to associate as well. They are infix (binary) and prefix (unary) as in C.

5. Using the unambiguous grammar for arithmetical expressions from your notes (Example 3.5 in CFG Introduction), draw a parse tree for: \((x+y)*x/z-x\).