

CS328, Fall 2002, Test 2

Time 70min. Use extra paper as needed, but make sure to identify each answer. Open notes. 5 questions, each 10 pct.

YOU MUST RETURN THIS PAGE. NAME _____

- 1 Design a DFA to recognize identifiers defined as follow:
 - identifier name can be any combination of letters and decimal digits as long as the total number of letters is greater than the total number of digits
 - identifier token may start with white spaces and it is always followed by one white space
 - the DFA should report whether the input is a valid identifier or error otherwise

Example:

x12y error
xy12z ok
12yyy ok

If you have done it, skip the rest. If you think this cannot be done, explain why and then assume that the total length of a token cannot be more than 3 characters (excluding white spaces) - redo the problem now or argue why it cannot be done.
- 2 Given the production:

S-> **aSAc** | **Acb**
A-> **bbb** | **empty**

implement a complete pseudocode for a recursive descent parser. Assume **scanner()** function returns the next token and **error()** aborting processing with an error message. Do not forget the main program. This grammar is LL(1) so no need to check nor modify.
- 3 Given

S -> **SabC** | **abC** | **aCa**
C -> **ccC** | **c** | **empty** | **D**
D -> **dd**

rewrite the grammar as LL(1) if possible or otherwise argue why it is not possible (and better have good arguments). **Prove** that it is indeed LL(1), showing **only** the sets that are needed and using them for your proof.
- 4 Suppose you have a language where a valid program is a sequence of statements and nothing else. Every statement ends with semicolon. A statement is either input **READ(variable)** or assignment **variable=expression**, where expression is C-like expression involving () and +,-,*,/ all arithmetical binary operators except - which is both unary and binary, and no other operators. Associativity is set so that all operators are left to right except * which is right to left, Precedence is set so that () overrides anything, the rest, from the strongest to the weakest are:
 - unary minus
 - + and -, the same
 - * and /, the same.

There are no numbers nor anything else. Variables are not defined.

Example program:

READ(x);

READ(y);

x=y+x/(x*y);

Design unambiguous CFG. Make sure to state what are the tokens.

- 5 Our project grammar uses input statement which reads input into a variable before it can be used. For example, to read a value and multiply by 10 we must do the following:

READ,x;

y = = x*10;

Suppose that we **also** want to allow the following:

y = = READ*10;

to do exactly the same semantics except the value is not stored in **x**. This change should apply to other cases as well, for example we want to be able to say:

IF (READ > 0) THEN ... #meaning if the input is greater than 0

y= =10+READ*READ ... # multiply two inputs, add 10, put result in y

Again, the original syntax/semantics should be preserved and this should be an additional way to accomplish something. Modify the original grammar to allow that. Show only the changes. For every changed production prove that it is still LL(1).

- 6 **Extra Credit:**

3pct) Using our language for the project, write a valid program that would read an input and then compute and output its factorial.

3pct) What are limitations of this program? Be very specific.