Programming Assignment
Lexical Analyzer Generator

It's required to design and implement a software tool that automatically constructs a lexical analyzer from a regular expression description of a set of tokens. In particular, the specification of the lexical analyzer is of the form:

```
P1      {Action 1}
P2      {Action 2}
.
.
Pn      {Action n}
```

Where each pattern Pi is a regular expression and each action (Action I) is a program fragment that is to be executed whenever a string matched by Pi is found in the input.

The tool is required to construct λ-accepters for the given regular expressions, combine these λ-accepters together with a new starting state, convert the resulting NDFSA to a DFSA, minimize it and emit the transition table for the reduced DFSA together with a lexical analyzer program that simulates the resulting DFSA machine.

The generated lexical analyzer has to read its input one character at a time, until it finds the longest prefix of the input, which matches one of the given patterns. If more that one pattern matches some longest prefix of the input, the lexical analyzer will break the tie in favor of the pattern listed first in the regular specifications. If none of the patterns matches any input prefix, an error recovery routine is called to print an error message and to continue to look for tokens. The lexical analyzer produces, as output, a stream of pairs in the form **<token, attribute value>.**

To test the lexical analyzer generator, use as input the regular definitions describing the set of tokens in the subset of PASCAL described in Appendix A of your text book sections A3, A4. To test the generated lexical analyzer, apply it to the simple test program given with the described language in Figure A1 of the same appendix.

General Notes:

1- Implement the project using the language of your choice
2- Each *group consists of 3* students
3- Each group must submit a project report. Make sure that your report contains the following:
   i) Pseudo-Code for the used algorithms.
   ii) A description of used data structures
   iii) The used example regular definitions, the resultant transition table for the minimal DFSA and the resultant stream of tokens for the example source program.
   iv) Any assumptions made and their justification.
   v) A simple user manual for the program
4- Each group must also submit a 3.5" diskette containing the following:
   i) The source code
   ii) The executable file. Please include any files needed by your program (e.g. DLL, BGI, etc.)

GOOD LUCK