COSC 5390
Lab Assignment II
DUE DATE: Thursday, 15 October 2015

Using the partially-completed Pascal procedure on pages 58-61 in your textbook (and the state table on page 56), write a complete lexical analyzer in the language of your choice that will identify tokens in a Pascal source program. This procedure (the scanner) could be called by the parser and would return (via the global variable *lex_state*) the state value of the token to the parser.

The input to your program will be a Pascal program. Output of your program should include:

- the input source program
- sequence of all tokens produced by the scanner (i.e. token type and corresponding lexeme)
- listing of the symbol table built by the program

Note that the program will require functions *kwsearch* (for searching a table loaded with the keywords on page 54) and *enter* (for entering identifiers in the symbol table). The symbol table should be constructed as a dynamic linked list or tree.

Also note that your lexical analyzer program should be run for the input Pascal source file listed on the following page.

Be sure to follow the techniques of good programming style and use extensive comments to provide for internal documentation of your source program(s). You will be required to submit *listings* of your source program file(s), your input data file, and your output files (multiple page listings should be individually stapled and with all listings clipped together). Graduate students are required to create at least one additional input source program for testing the scanner.

DATA FOR LAB ASSIGNMENT II:

```
program bin(input, output);
{this is a binary search routine}
type data = array[1..100] of integer;
var a:data; i,index,item:integer;

procedure binsrch(a:data; i,j:integer; var index:integer);
var mid:
    integer;
begin
if j < i then index:=-1   {search failed}
else begin
       mid:=(i+j) div 2;
       if a[mid] = item then index:=mid        {found it}
       else if a[mid]<item then binsrch(a,mid+1,j,item,index)
       else binsrch(a,i,mid-1,item,index)
     end;
end;

begin
for i := 1 to 100 do a[i] := 2*i+(i mod 2);
    for i := 1 to 5 do
    begin
      read(item);
      binsrch(a,1,2*i,item,index);
      writeln(item,index)
    end
end.
```