The Parser

Now that we have a READER that will read the input remove comments and store the input in a character array, we are ready to move on to the parser phase of the interpreter.

This assumes that you have all the input in an array (comments removed) say "userinput".

Implement the Following

0) Create a symbols table and a symbol table manager. To make life easy use pointers to characters, also read below for more clarification. We are not going to trees to implement this.

1) Write the function: int ParseName();

It should read characters and make them into a string. Then install it in the symbol table and return the location at which the string was stored.

If it is already in the symbol table return the location where it was found.

2) Write the function NAMELIST ParseNameList():

This function return a pointer to a list of names by doing the following:

It should repeatedly call ParseName() (ParseName() will return an integer after installing the name in the symbol table). Then make a linked list containing those names (integers).

So NAMELIST is a pointer to the head of the linked list that contains names (int's).

3) Write the function int ParseDef:

This returns an integer location where the function name is stored in the symbol table.

ParseDef: This is a little bit more complicated, will be explained in class.

4) Write the function: void NewfunDef();
(this manages the linked list of functions)

This should install a function record in the linked list pointed to by
the global variable "fundefs"(or whatever you want to call it).

function record should have the following fields:

a) name (int location of the function name in the symbol table)
b) arglist set to the pointer value returned by ParseNameList() .
c) body; this is a pointer to an exp.
d) nextptr; which can point to another function in the linked list.

4) int skipblanks(pos) //given in class

pos is an integer variable which starts off at the zeroth position of the userinput. When called, will skip blanks and returns the current position where the input in not a blank. So userinput[pos] will be non blank.

THE NEXT TWO FUNCTIONS ARE EASILY IMPLEMENTABLE IF YOU USE MUTUAL RECURSION:

5) EXP ParseExp(): This should build an expression tree and return a pointer to it. (EXP is same as EXPREC*)

6) EXPLIST ParseExpList(): This should repeatedly call parseExp() and return a pointer to a linked list of expressions. (EXPLIST is the same as EXPLISTREC*)

4) Write a function NUMBER parseVal(): This takes a sequence of integers possibly proceeded by a minus sign. Then convert it to an integer and return it. (NUMBER is the same as int)

COME TO CLASS FOR MORE EXPLANATION AND HELP.