# Pseudocode Design

# Pseudocode for main:

*Function declaration*:

**int** **main**(**int** argc, **const** **char** \* argv[])

*Inputs/Outputs:*

Input fileNEWTON.PAS

Integer returns value as error indicator

Output to screen each line in source file and tokens

*Pseudocode*:

Open the input file NEWTON.PAS.

Allocate memory space for the token list

Initialize the token list to point to nothing

Loop as long as the token string is not a period ‘.’ (end of Pascal program)

Get a new token

Add the new token to the linked list

Print the token to the screen

# Pseudocode for quit\_scanner:

*Function declaration*:

**void** **quit\_scanner**(FILE \*src\_file, Token \*list)

*Inputs/Outputs:*

Pointer to file

Pointer to token list

*Pseudocode*:

Close the input file

Loop through the list of tokens

De-allocate memory for each token

# Pseudocode for add\_token\_to\_list:

*Function declaration*:

**void** **add\_token\_to\_list**(Token \*list, Token \*new\_token)

*Inputs/Outputs:*

Pointer to list of token

Pointer to new token

*Pseudocode*:

Add the new token to the beginning of the linked list

# Pseudocode for init\_lister:

*Function declaration*:

FILE \***init\_lister**(**const** **char** \*name, **char** source\_file\_name[], **char** dte[])

*Inputs/Outputs:*

Pointer to a file name by ref

Array of date by ref

Return file pointer

*Pseudocode*:

Open the file of the given name

Get the date time

# Pseudocode for downshift\_word:

*Function declaration*:

**static** **char** \* **downshift\_word**(**char** token\_string[])

*Inputs/Outputs:*

String to be converting to lower case

Return pointer to the converted string

*Pseudocode*:

Loop through each letter in the string and convert it to lower case

Return the pointer to the converted string

# Pseudocode for is\_reserved\_word:

*Function declaration*:

**static** BOOLEAN **is\_reserved\_word**(**char** token\_string[], Token \*token)

*Inputs/Outputs:*

String to check if is a reserved word or just an identifier

Token pointer to update the token code

*Pseudocode*:

Loop the symbol string array to find if the given string is in the table. It yes then the code is a reserved word or else is an identifier

# Pseudocode for get\_special:

*Function declaration*:

**static** Token \***get\_special**(**char** token\_string[], Token \*token2)

*Inputs/Outputs:*

A string of special symbol

A token pointer

Return token pointer

*Pseudocode*:

Loop through the symbol sting array

Compare each array element to the token sting to identify the token code

Update the token code

Return the token pointer to the caller

# Pseudocode for get\_string:

*Function declaration*:

**static** Token \***get\_string**(**char** token\_string[], Token \*token2)

*Inputs/Outputs:*

Token string

Pointer to token

*Pseudocode*:

Update the token string with the input string

Update the token code with the token code enum

Update the token type with the string literal enum

Update the token pointer to null

Return the token pointer

# Pseudocode for get\_number:

*Function declaration*:

**static** Token \***get\_number**(**char** token\_string[], Token \* token2)

*Inputs/Outputs:*

Token string

Token pointer

*Pseudocode*:

Update token code to number enum

Update token type to integer literal enum

Initialize token next pointer to null

Copy the token string to token

Return updated token pointer

# Pseudocode for get\_word:

*Function declaration*:

**static** Token \***get\_word**(**char** token\_string[], Token \*token2)

*Inputs/Outputs:*

Token string

Token pointer

*Pseudocode*:

Convert the token sting to lower case

Update token next pointer to null

Update token type to string literal enum

Copy the token string to the token

Update token code to either reserve word or identifier

Return the token pointer to caller

# Pseudocode for skip\_blanks:

*Function declaration*:

**static** size\_t **skip\_blanks**(**char** source\_buffer[], size\_t j)

*Inputs/Outputs:*

String to be manipulated

Current index pointing to a character in string

Return index to non-blank character

*Pseudocode*:

Loop through the string start from current index passed in

If the character is blank the go to next character

If the character is not blank then return the index

# Pseudocode for skip comment:

*Function declaration*:

**static** size\_t **skip\_comment**(**char** source\_buffer[], size\_t j)

*Inputs/Outputs:*

*String to be manipulated*

*Current index pointer that point to a character in the string*

Return the index point past comment

*Pseudocode*:

Loop through the string from the current index pointer

Check for beginning Pascal comment “{“

If found then skip past ending comment symbol “}”

Return index after comment

# Pseudocode for get\_char:

*Function declaration*:

**static** **char** **get\_char**(**char** token\_string[])

*Inputs/Outputs:*

*Token string*

*Return the first character in the token string*

Initialize a static index to keep track of the current line pointer

*Pseudocode*:

Extract the first character in the string

If it is end of line or null character then

Get a new string from the source file

If there is nothing in the source file then return eof

Else

Return the first character that is not a comment or blank

Pass the sting and the current index to function buildToken to get the next index

Return the first non-blank character