PROGRAMMING LANGUAGES HOMEWORK #1

Project Overview

This project includes a set of Lisp functions that aim to convert lines of code in the C language to the Common Lisp format.

Functions

1. **(defun line-type (line)**

   Purpose: Determines what type of line it is (for example, if statement, for loop, etc.).

   Example:

   Input: "for (int i = 0; i < 10; i++) {"
   Output: 'for-loop

2. **(defun conversion-foo (line)**

   Purpose: Uses line-type to find the line type and sends it to the correct conversion function based on that type.

   Example:

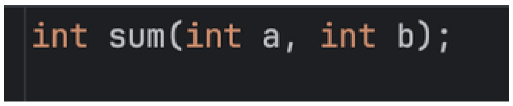
   Input: for-loop
   Output: convert-for-loop

3. **(defun convert-function-declaration (line)**

   Purpose: Converts a C function declaration to a Lisp declaim format.

   Example:

   Input: "int sum(int a, int b);"
   Output: (declaim (ftype (function (integer integer) integer) sum))



```
int sum(int a, int b);
```

```
(declaim (ftype (function (integer integer) integer) sum))
```

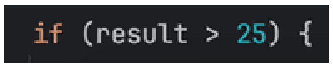
4. **(defun convert-if-statement (line)**

   Purpose: Converts an if statement to Lisp format by extracting the variable, operator, and value.

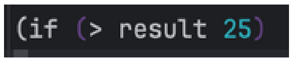
   Example:

   Input: " "if (result > 25) {"

   Output: (if (> result 25)



```
if (result > 25) {
```



```
(if (> result 25)
```

5. **(defun function-call (line)**

   Purpose: Checks if a line is a function call in C.

   Example:

   Input: "result = sum(a, b);"

   Output: t (indicating it's a function call)

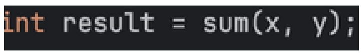
6. **(defun convert-function-call (line)**

   Purpose: Converts a function call in C to Lisp setf format.

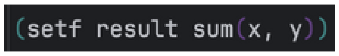
   Example:

   Input: "result = sum(a, b);"

   Output: (setf result sum(a, b))



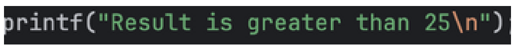
```
int result = sum(x, y);
```



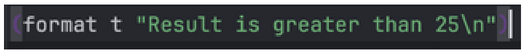
```
(setf result sum(x, y))
```

7. **(defun convert-printf-statement (line)**

   Purpose: Converts printf statements to Lisp format statements.



```
printf("Result is greater than 25\n")
```



```
(format t "Result is greater than 25\n")
```

   Example:

   Input: "printf("Result is greater than 25\n");"

   Output: (format t "Result is greater than 25~%")

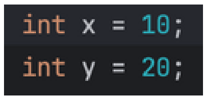
8. **(defun convert-variable-declaration (line)**

   Purpose: Converts variable declarations in C to Lisp setf format.

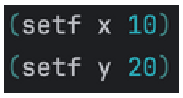
   Example:

   Input: "int x = 20;"

   Output: (setf x 20)



```
int x = 10;
int y = 20;
```



```
(setf x 10)
(setf y 20)
```

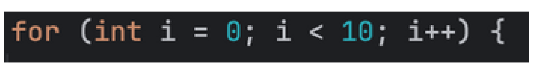
9. **(defun convert-for-loop (line)**

   Purpose: Converts C for loops to Lisp loop format.

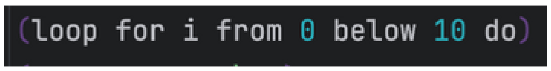
   Example:

   Input: "for (int i = 0; i < 10; i++) {"

   Output: ((loop for i from 0 below 10 do)



```
for (int i = 0; i < 10; i++) {
```



```
(loop for i from 0 below 10 do)
```

10. **(defun convert-return-statement (line)**

Purpose: Converts C return statements to Lisp expressions.

Example:

Input: "return a + b;"
Output: (+ a b)

```
return a + b;
```

```
(+ a b)
```

11. **(defun convert-type (c-type)**

Purpose: Converts C types like int to Lisp types like integer.Example:

Example: Input: "int"
Output: "integer"

12. **(defun convert-function-definition (line)**

Purpose: Converts a C function definition to a Lisp defun function.

Example: Input: "int sum(int a, int b) {"
Output: (defun sum (a b))

```
int sum(int a, int b) {
```

```
(defun sum (a b))
```

13. **(defun convert-line (lines)**

Purpose: Finds the line type and sends it to the right conversion function.

14. **(defun read-file (filename)**

Purpose: Reads each line of a file into a list of lines.

15. **(defun write-file (filename lines)**

Purpose: Writes the converted lines to an output file.

16. **(defun main ()**

Purpose: Main function that reads, converts, and writes lines to the output file.