Evan Bourgoine (ebourgoine25)

CS3304 Project 2 Description & Documentation

Data Structures Used to Store Data and Functions:

1. Sample List:

The sample list is used to store the processed temperature readings as a list of numbers in Fahrenheit.

For example: '(273.15, 373.15, 32.00, etc.)'

The operations for this data structure were to be able to add new samples to the head of the list, clear the list to set it back to an empty list, '()', and display the list in the order it was given in the file.

2. Symbol Table:

The symbol table data structure was used to store the user-defined post-processing functions, such as 'proc1 '(* 2 x)' , etc.

The operations for this data structure were to add new key-value pairs, remove them, and display them, similar to the sample list, but using key-value pairs to be able to match the given function name with the operation.

3. The post-processing function

The post-processing function can be considered a data structure as it had to store the current lambda function that would be applied to the readings.

For example, '(lambda (x) (* 2 x))'

The operations for this would be to update and clear the active function.

How is User Input Processed?

The program processes input line by line per the instructions, and it categorizes each line based on the prefix. I created a process-line function to be able to update the state of the program. The convert-to-fahrenheit function was also created to take the lines which have prefixes cC, fF, or kK and convert them to Fahrenheit. The program also takes lines without any prefix to be Fahrenheit by default as seen in the example input/output files given.

How is Data Displayed?

I used the function printf to display the data to the user to match the expected output.

When I came across an invalid input, I would have the program ignore it and logged accordingly.

Conclusion:

The Scheme program created uses the samples list, symbol table, and post-processing function by categorizing the input and performing the updates needed. My overall approach to this project was to work from the project instructions, such as starting with the basic instructions until it worked correctly from the input/output files given, and then move to the post-processing section, and finally the editing requirement. When I would run into issues with the output not being what was expected, I would primary add more printf statements at different areas of the code to check if the data was being stored, and processed correctly.