Assignment 1

CSI 3120

Author: James Dickens

Email: jdick088@uottawa.ca

Assignment 1: Lambda Calculus Syntax

Program Requirements:

- Input a text file, containing a new input string per line
- Perform lexical analysis to parse each string to tokens
- Two outputs considered:

  1: Determine the input string is valid, and if it is not print the error

  2. For valid input strings, output a parse tree

Grammar

```
<expr> ::= <var> | '(' <expr> ')' | '\' <var> <expr> | <expr> <expr>
```

With the following constraints:

- Variables (var) should begin with a letter (either uppercase or lowercase), followed by strictly alphanumeric characters.
- Slashes should be directly followed by a variable name
- For lambda expressions (column 3 in the grammar), the dot operator is allowed, where
- \x.x b is equivalent to \x (x b)
- The dot operator must be preceded by a variable, and the next token must be either a parenthesis, or a variable name, i.e. \x.(x b) is also acceptable and equivalent to \x (x b)
- The final parsed string should replace dot operators with parentheses as part of the parsing process
- as in \x.\y.x y becomes \x(\y(x y)

Parsed Output String

- The final parsed output string consists of a list of tokens, where each token is either a variable name, an opening parenthesis (, a closing parenthesis), or a slash.
- Example, using _ as a delimiter for printing, \x.(x za) -> _x_(_x_za_)

Or in list of strings form, ["\", "x", "(", "x", "za", ")"]

Error Types

- For invalid input strings, the error type and starting index of the string where the error occurs should be printed to the console.
- Consider these errors:
- Invalid character (ex. +, -, etc.)
- Invalid spaces in lambda expressions, incorrect lambda syntax
- Missing expressions (ex. \x())
- Non-matching brackets (ex. (a)))
- Invalid variable names

See the sample output, *valid_examples.txt invalid_examples.txt* for examples.

Parse Trees

- For the second output type, parse trees should be printed to the console for valid input strings
- Example, for the input string \x (\y (x y)), delimiting tokens with _, we would have:
- _x_(__y_(_x_y_)_)
- ----\
- ----x
- ----(
- ----_y_(_x_y_)
- --------\
- --------y
- --------(
- --------x_y
- ------------x
- ------------y
- --------)
- ----)

Code

- Written in Python, recommended version 3.5+ since the code uses type hints.
- Use only standard library functions, no additional libraries
- Complete all functions with *#TODO* in A1.py, some are marked with the additional keyword Optional, these are to be completed for bonus marks)

Important Functions

read_lines_from_txt(fp: [str, os.PathLike]) -> List[str]:

is_valid_var_name(s: str) -> bool:

parse_tokens(s_: str) -> Union[List[str], bool]:

read_lines_from_txt_check_validity(fp: [str, os.PathLike]) -> None:

read_lines_from_txt_output_parse_tree(fp: [str, os.PathLike]) -> None:

build_parse_tree_rec(tokens: List[str], node: Optional[Node] = None) -> Node:

build_parse_tree(tokens: List[str]) -> ParseTree:

Optional: Add Associativity

- Add brackets to tokenized strings to remove ambiguity, i.e.
- Left associativity to (a b c) would yield ((a b) c),
- And right associativity to (a b c) would yield ((a) (b c))

See

add_associativity(s_: List[str], association_type: str = "left") -> List[str]:

To Submit

- Include a README file with:
  - Student numbers.
  - Information on whether the program works or has known defects.
  - Any deviations from the assignment requirements, if applicable.
  - Reference all the websites and external sources you used for help, so you don’t fall into plagiarism.
  - If you used any AI tool for help, like ChatGPT, you need to screenshot the whole chat and include it in the .zip file.
- Include your well commented code (A1.py). Use the template provided. You can add functions to the template.
- Include Four .txt files:
  - Valid examples given to you,
  - Negative example given to you
  - Extra valid examples you chose. Please use file path name: *extra_valid_examples.txt*
  - Extra negative examples you chose. Please use file path name: *extra_invalid_examples.txt*