**COVER PAGE**

# **CS323 Programming Assignments**

**Fill out all entries 1 - 7. If not, there will be deductions!**

**Check one**

1. Names [ 1. John Tu ], (MW [ X ] or R class [ ] )

[ 2. ], (MW [ ] or R class [ ] )

[ **if 3**. ], (MW [ ] or R class [ ] )

2. Assignment Number [ 2 ]

3. Due Dates **Softcopy**  [ 5/10 ], **Hardcopy** [ 5/10 ]

4. Turn-In Dates **Softcopy** [ 5/10 ], **Hardcopy** [ 5/10 ]

5. Executable File Name [ Intermediate.exe ]

(**A file that can be executed without compilation by the instructor**)

6. Lab Room [ CS 104 ]

**(Execute your program in a lab in the CS building before submission)**

7. Operating System [ Windows 10 ]

**To be filled out by the Instructor:**

GRADE:

COMMENTS:

Documentation #3

1: This program will implement a syntax analysis procedure to determine whether the source code contains valid syntax. In order to do so, any of the top-down parser techniques will be used: recursive descent parser, predictive recursive descent parser, or table driven predictive parser.

2: How will the syntax analyzer be implemented in C++ code? As mentioned in the previous assignments, the procedure will remain similar regardless of the changes made to the source code. The syntax analysis program consists of header files that contains the function prototypes and the variables needed to read in the data and the .cpp files that holds the function declarations along with the main function acting as the driver. First, the text file will be read as an input file, and then for each line in the text file, there will be reading in each character of the string and call the appropriate functions to categorize them by their type. The process continues on until the end of file is reached, in which the analyzer will print out the results on console and on another text file. Once that is done, an additional step will be conducted, and that is to do the grammatical parsing based on the rules provided. Reading through each line of output, the Boolean functions will determine where the lexeme belongs to and the statements will be printed out if the input satisfies the following conditions. After the grammatical parsing is completed, the results will be printed on the console window for the user to read.

3: What are the main features included in doing the syntax analysis? In this program, the syntax analysis source code follows the same procedure as the lexical analysis, except that an additional step is performed once the categorization of the tokens is processed. There are several Boolean functions that takes the index as a reference value. In each declaration of the function, it checks each lexeme on which type it belongs to and depending on whether the conditions are met, the function either prints out the statement of the result and returns true if all the type requirements are satisfied or false if not. Not only that, prior to the conditional check, if the index of the input token exceeds the index for the lexemes, return false to indicate the current index exceeded the maximum number. There will be Boolean variables assigned as ‘flags’ that calls each appropriate Boolean function and the results returned will be stored in those variables when performing the condition check of the lexeme types. In addition to the aforementioned Boolean functions mentioned above, treat the lines containing the exclamation marks as the comment, in which we would like to ignore them. Finally, since the pass-by-reference value cannot be modified directly, a temporary variable is declared and assigned to the token index, and it will be updated according to the result of the Boolean function. The third assignment contains all of the following declarations as mentioned in the second assignment, except that additional flags are implemented in the program, such as the if-else statements and the while loops.

4: Are there any limitations that exist in the source code? If so, then as mentioned before in the previous assignments, I always define the array for the number of tokens, the token types, and token line numbers with 1000 elements as a maximum limit because I am concerned about the consumption of excessive dynamic memory. Also, the sample input files provided are reused from the lexical analysis because it would be time-consuming to write down a new source code from an empty text file.

5: Perhaps are there any shortcomings that exist when compiling and putting the program together? If so, then I was about to generate an assembly code listing of the sample source code along with constructing a symbol table handling in order to store the lexemes and a memory address for the identifier. Given that I am running short of time in this semester and I know little about intermediate code generation, I decided to omit these required components because it will be very time consuming to declare those functions above.