Sairam Soundararajan

CMSC430: Advanced Programming Languages

University of Maryland Global Campus

Professor Jarc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Expected Output | Actual Output | Pass? |
| 1 | ./compile < test4.txt | $ ./compile < test4.txt  1 -- Program containing arrow symbol and new reserved words  2  3 function main b: integer returns integer;  4 a: real is 3;  5 begin  6 if a < 2 then  7 7 + 2 \* (2 + 4);  8 else  9 case b is  10 when 1 => a \* 2;  11 when 2 => a + 5;  12 others => a + 4;  13 endcase;  14 endif;  15 end; |  | YES |
| 2 | ./compile < test5.txt | $ ./compile < test5.txt  1 -- Program containing the new operators  2  3 function main b: integer, c: integer returns integer;  4 a: real is 3;  5 begin  6 if (a < 2) or (a > 0) and (b /= 0) then  7 7 - 2 / (9 rem 4);  8 else  9 if b >= 2 or b <= 6 and not(c = 1) then  10 7 + 2 \* (2 + 4);  11 else  12 a \*\* 2;  13 endif;  14 endif;  15 end; |  | Yes |
| 3 | ./compile < test6.txt | $ ./compile < test6.txt  1 -- Program containing the comment, modified identifer and new literals  2  3 // This is the new style comment  4  5 function main b: integer, c: integer returns real;  6 a: real is 3.;  7 d: real is 5.7;  8 a\_1: real is 4.e2;  9 ab\_c\_d: real is 4.3E+1;  10 ab1\_cd2: real is 4.e-1;  11 begin  12 if (a < 2) or (a > 0) and (b /= 0) or false then  13 7 - 2 / (9 rem 4);  14 else  15 if b >= 2 or b <= 6 and not(c = 1) and true then  16 a\_1 + ab\_c\_d \* (ab1\_cd2 + 4);  17 else  18 a \*\* 2;  19 endif;  20 endif;  21 end; |  | Yes |

Snapshots of Test Run:

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

Text

Description automatically generated

Reflection:

For this project, I used notepad for editing and Cygwin for building, compiling, and testing. This is the first time I’ve ever worked with compiler design and lexical analyzers so I was struggling to understand what to do at first. I started off by using Ubuntu and I realized that the program wasn’t going to work as good as Cygwin. So, I switched to Cygwin before I actually began working on the project requirements. After installing the necessary plugins for Cygwin, I was able to build and compile the project. I did have some struggles as to what the instructions asked for, but with the professor’s help, tutoring I was lucky to receive, lecture videos, and the textbook, I was able to succeed in editing the lexical analyzer, tokens header, and listing compiler code accordingly. For the most part, I worked on adding more reserved words with their respective translation rules and regular expressions. Working on the listing code was a refresher for me, as I have not worked with C++ in a long time. Even though I needed more time on this project, I was fortunately able to succeed with the help that I received from the professor and other resources. If there is anything I could do to plan my time better, I’ll try to work backwards in order to spend a good amount of time on every part of the projects.