# **COMPILER DESIGN CLASS PROJECT – OPERATOR PRECEDENCE PARSER**

## **EXPLAINATION**

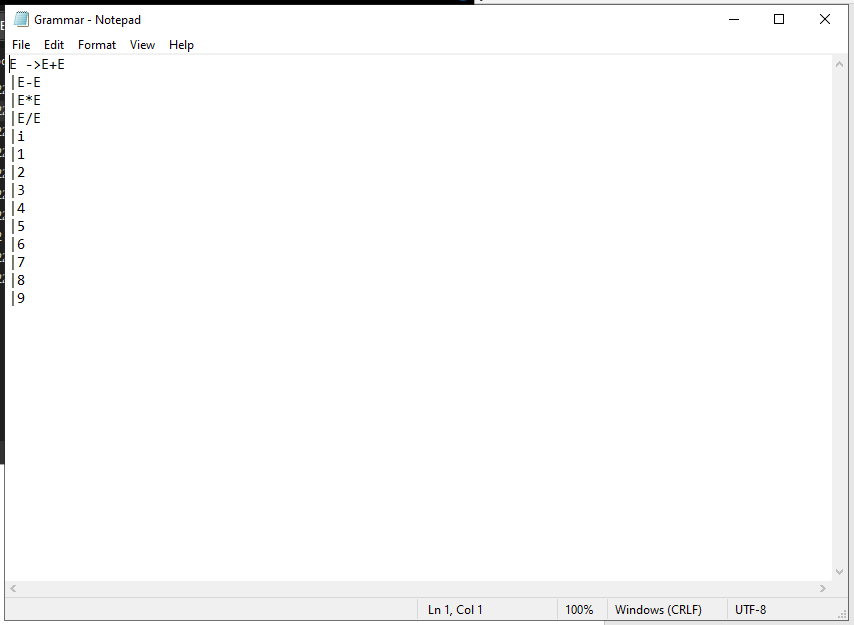
It is a type of shift reduce parser, but in precedence operator, precedence parser, as it’s clear by its name it’s used for the parsing of the statements having some kind of operators as athematic operator.

Grammar is set to be operator precedence if it has following two properties

* No right-hand side of any production has sigma or epsilon
* No two none terminals are adjacent

## **GRAMMAR**

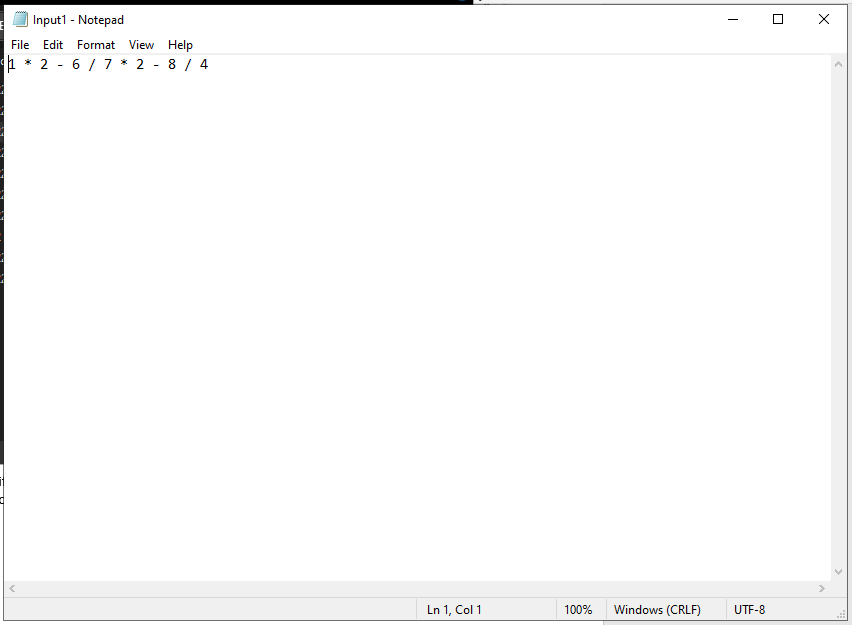
Following is the grammar where the inputs are from one to nine. Following both of the properties of operator precedence parser that’s no signa or epsilon sign can be seen at the right side of any production rule below additionally there are no adjacent non terminals.



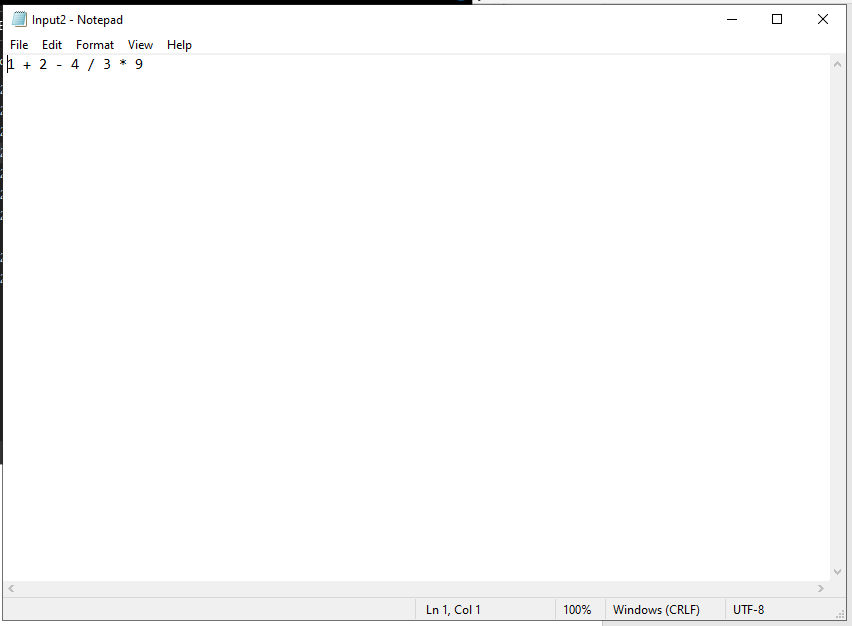
## **INPUT**

Following are the different inputs that got tested in this project, two inputs are correct whereas the rest of the three are incorrect.

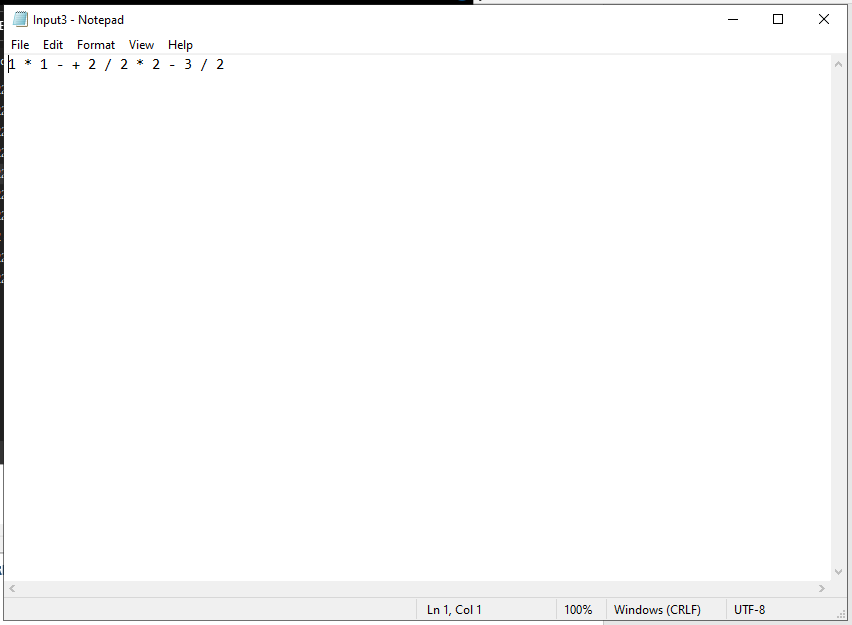
### **INPUT 1 - CORRECT**



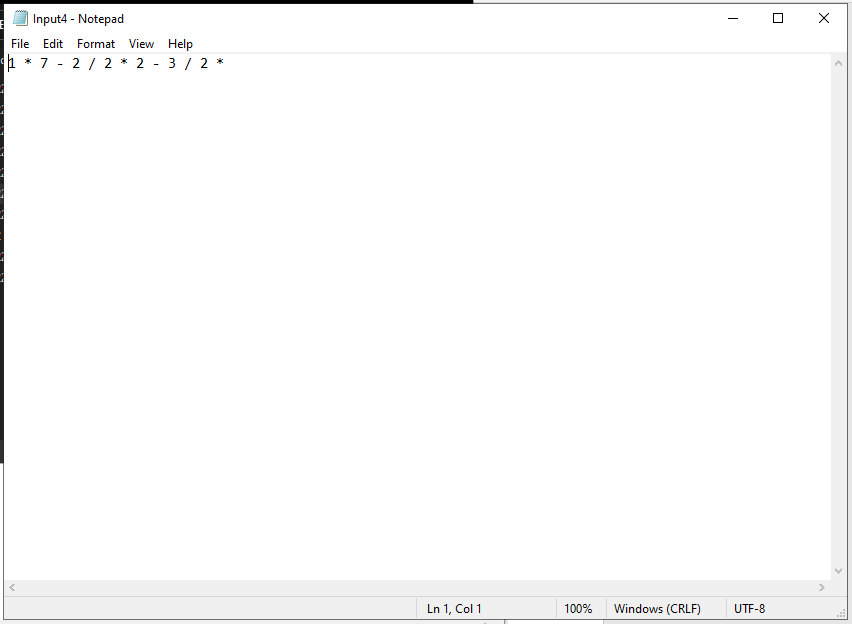
### **INPUT 2 - CORRECT**



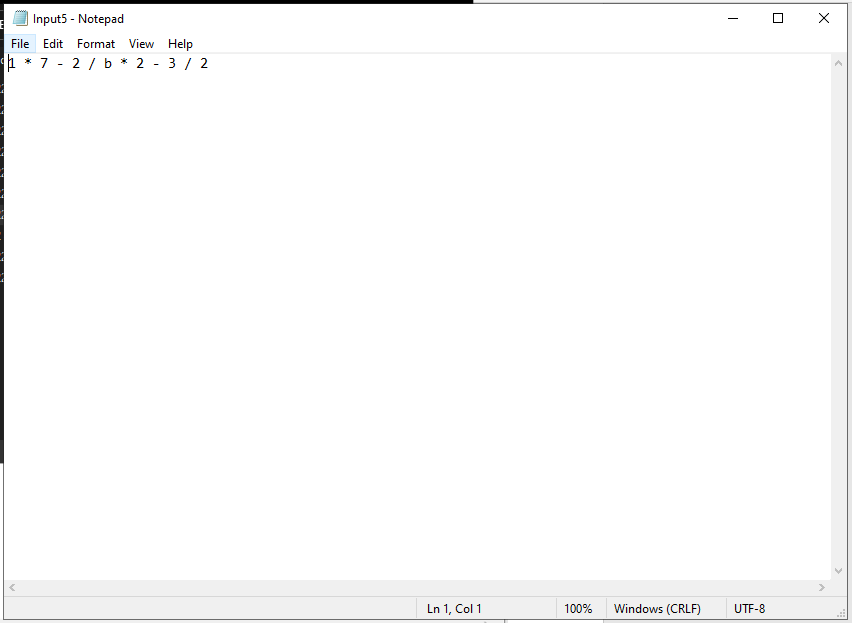
### **INPUT 3 – INCORRECT (TWO OPERATORS APPEAR TOGETHER)**



### **INPUT 4 – INCORRECT (OPERATOR AT THE VERY END OF A STATEMENT)**

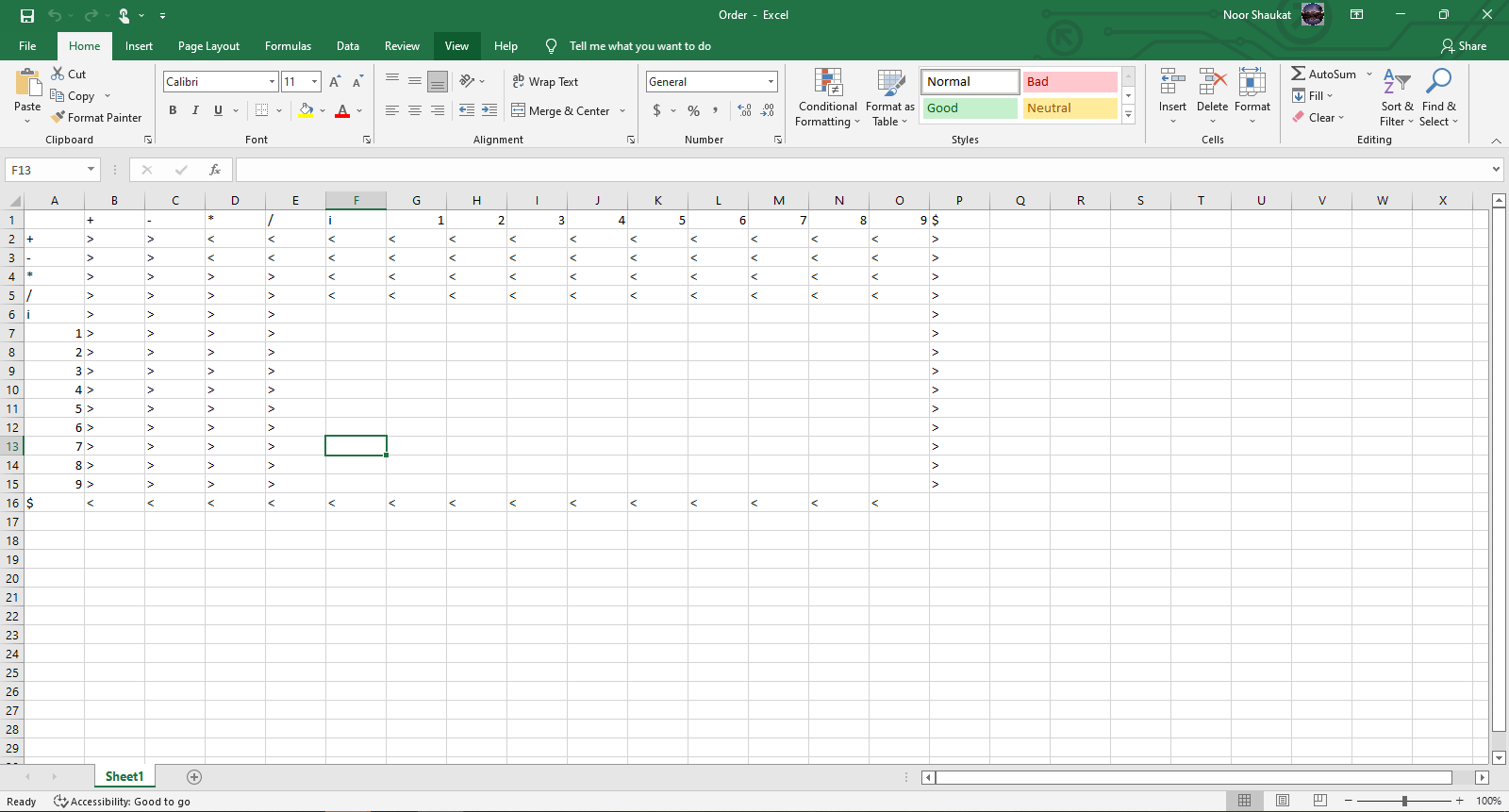


### **INPUT 5 – INCORRECT (UNDEFINED CHARACTER)**

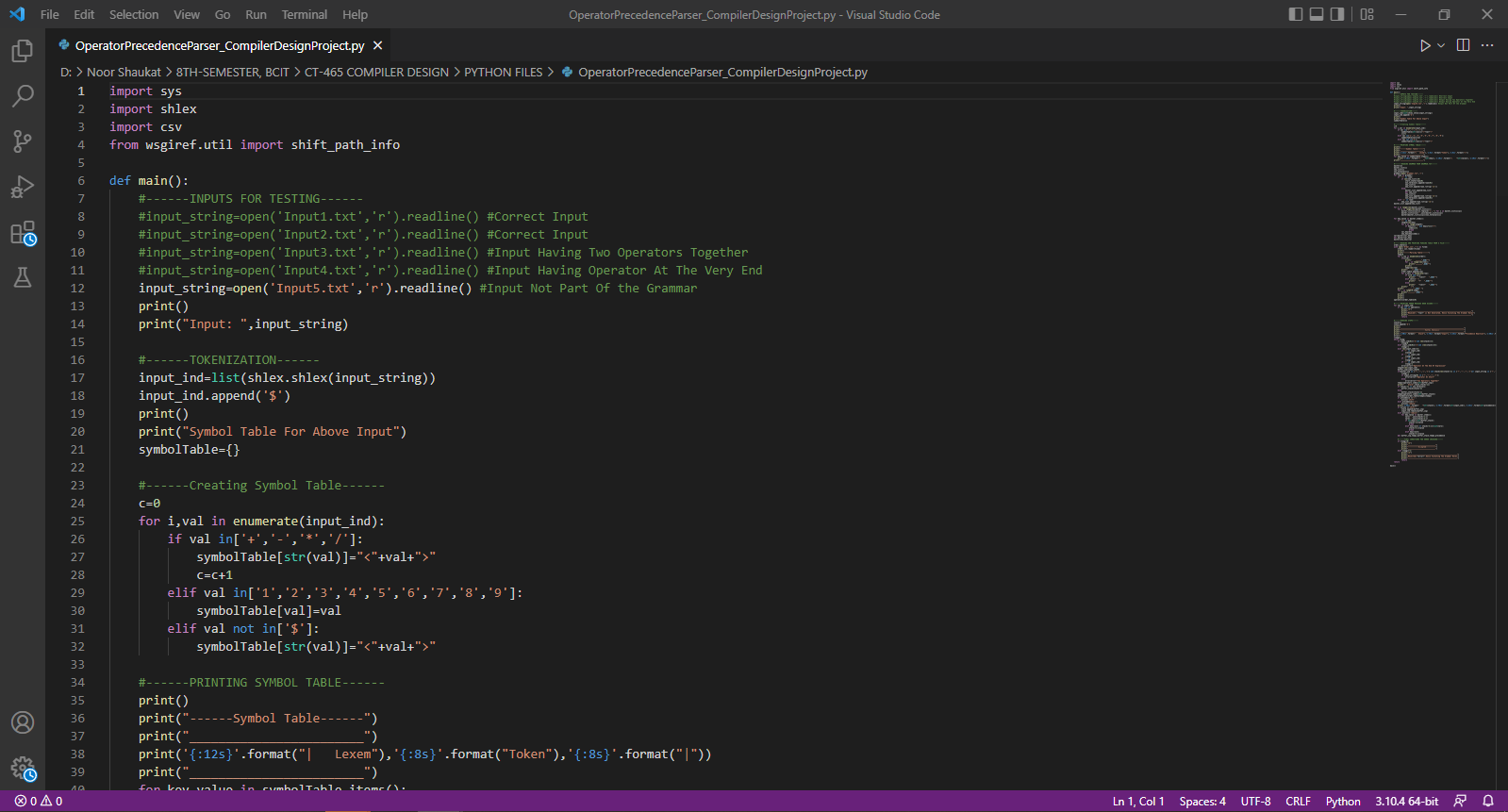


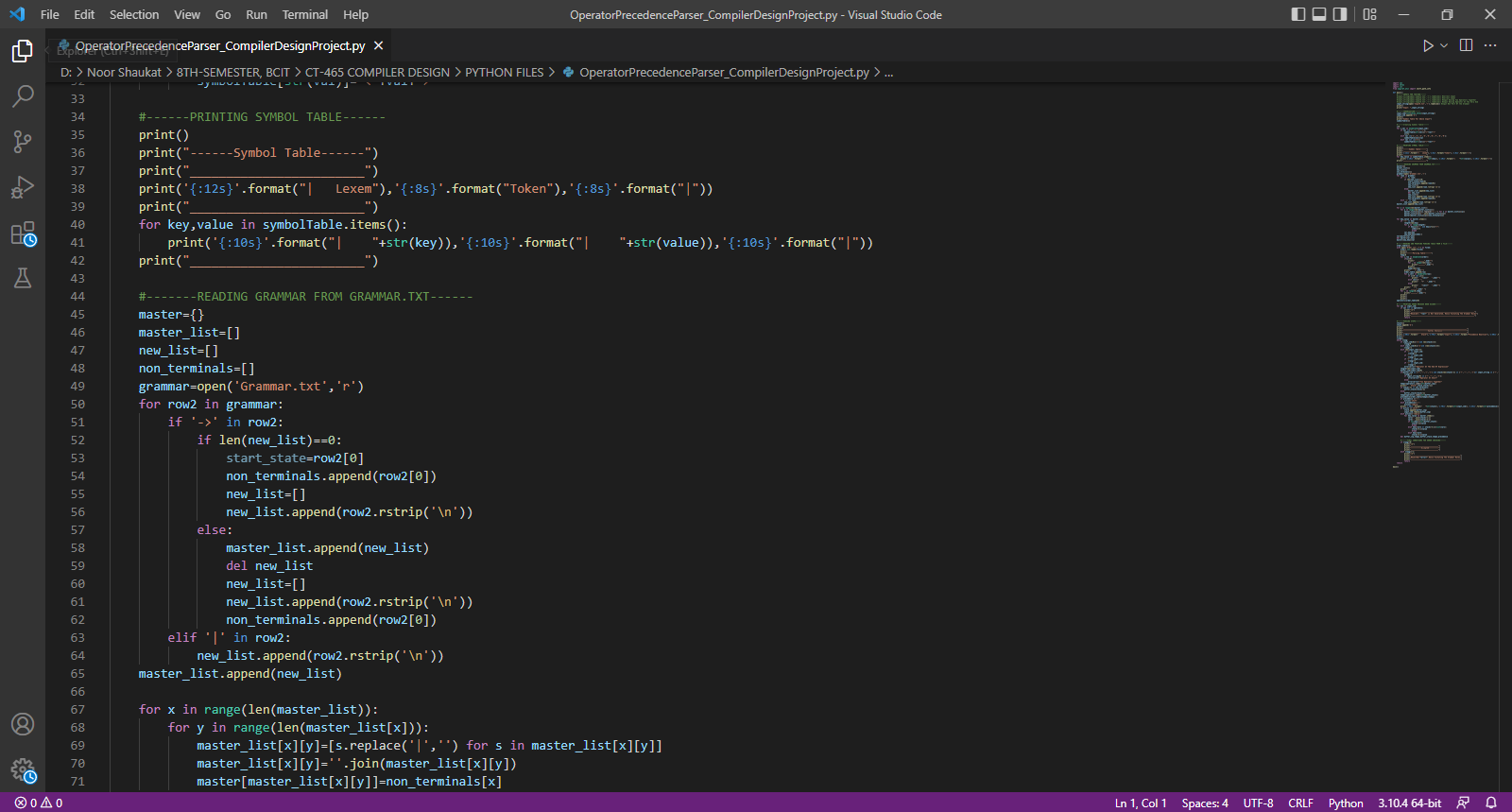
## **PARSRING TABLE**

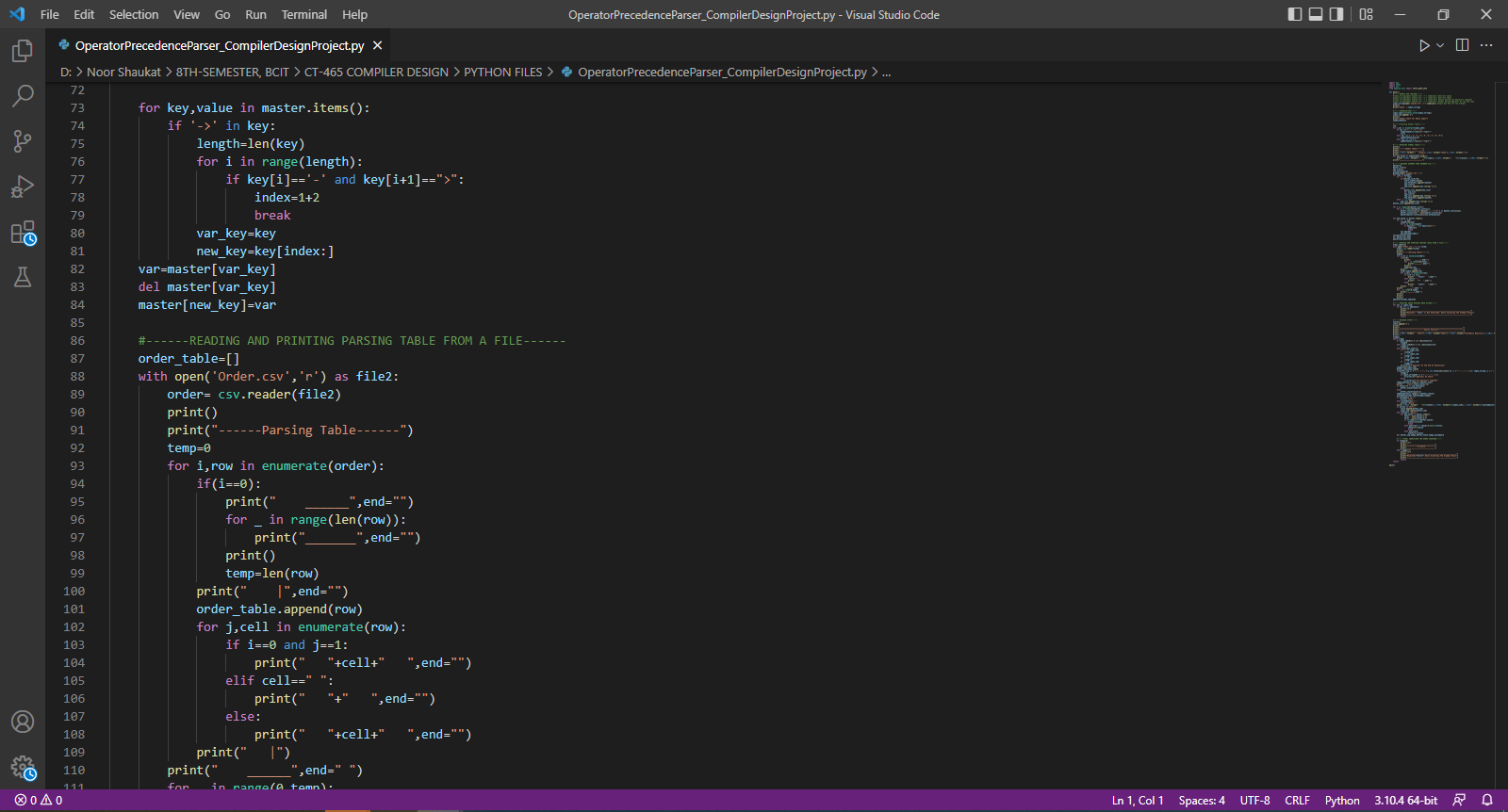
All the written rules of the grammar will be followed by this parsing table.

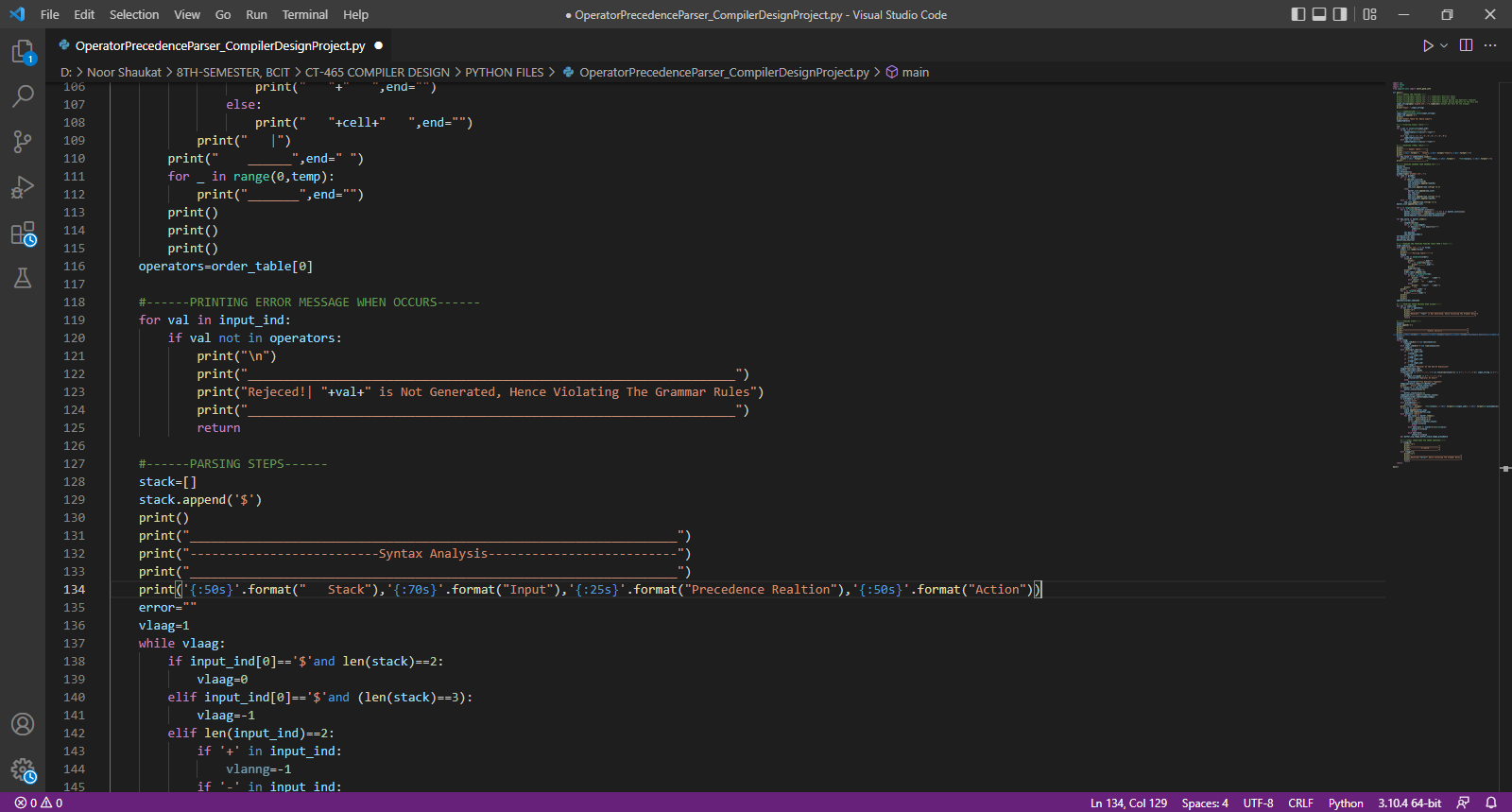


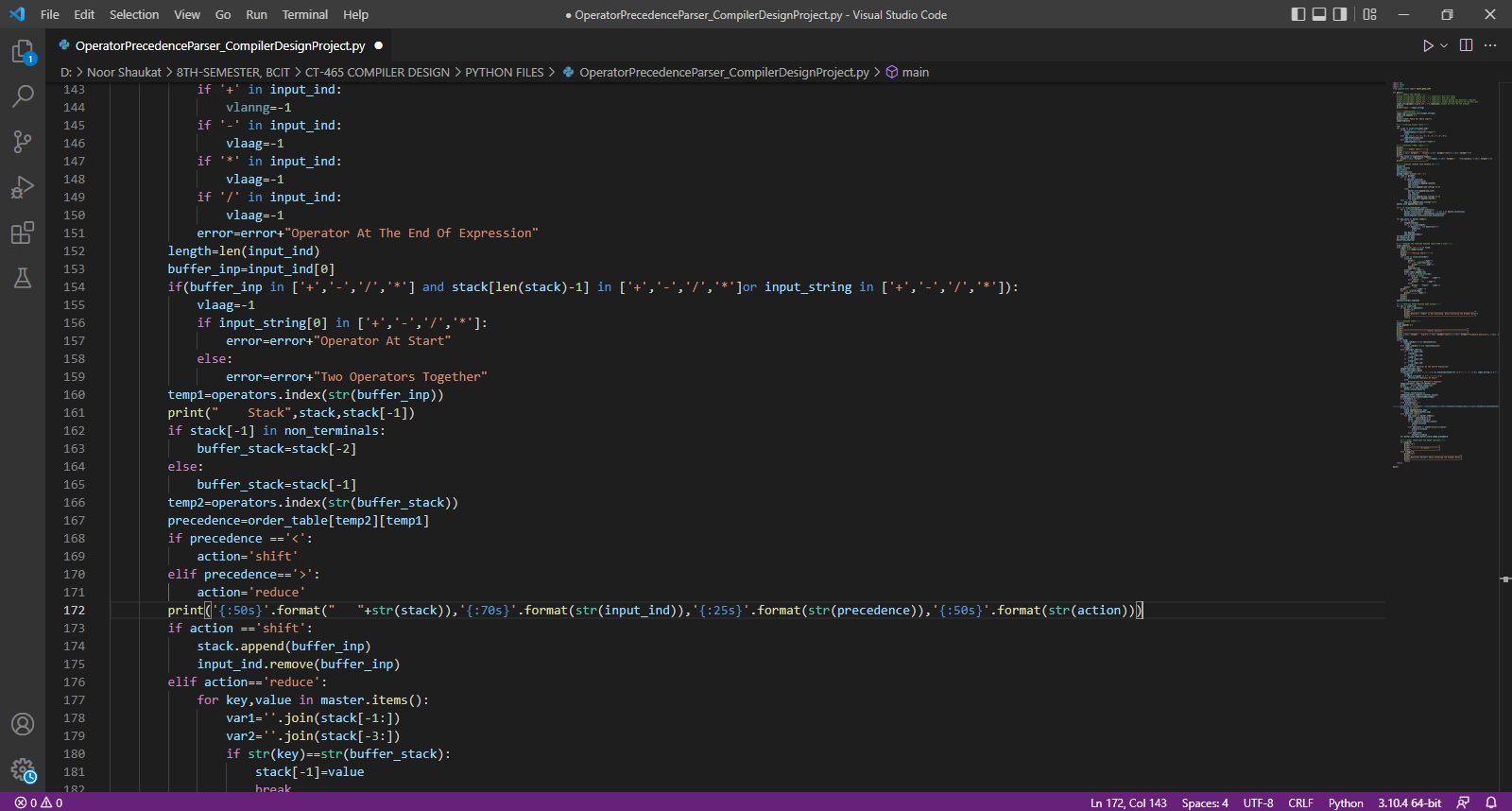
## **CODE DEMOSTRATION**

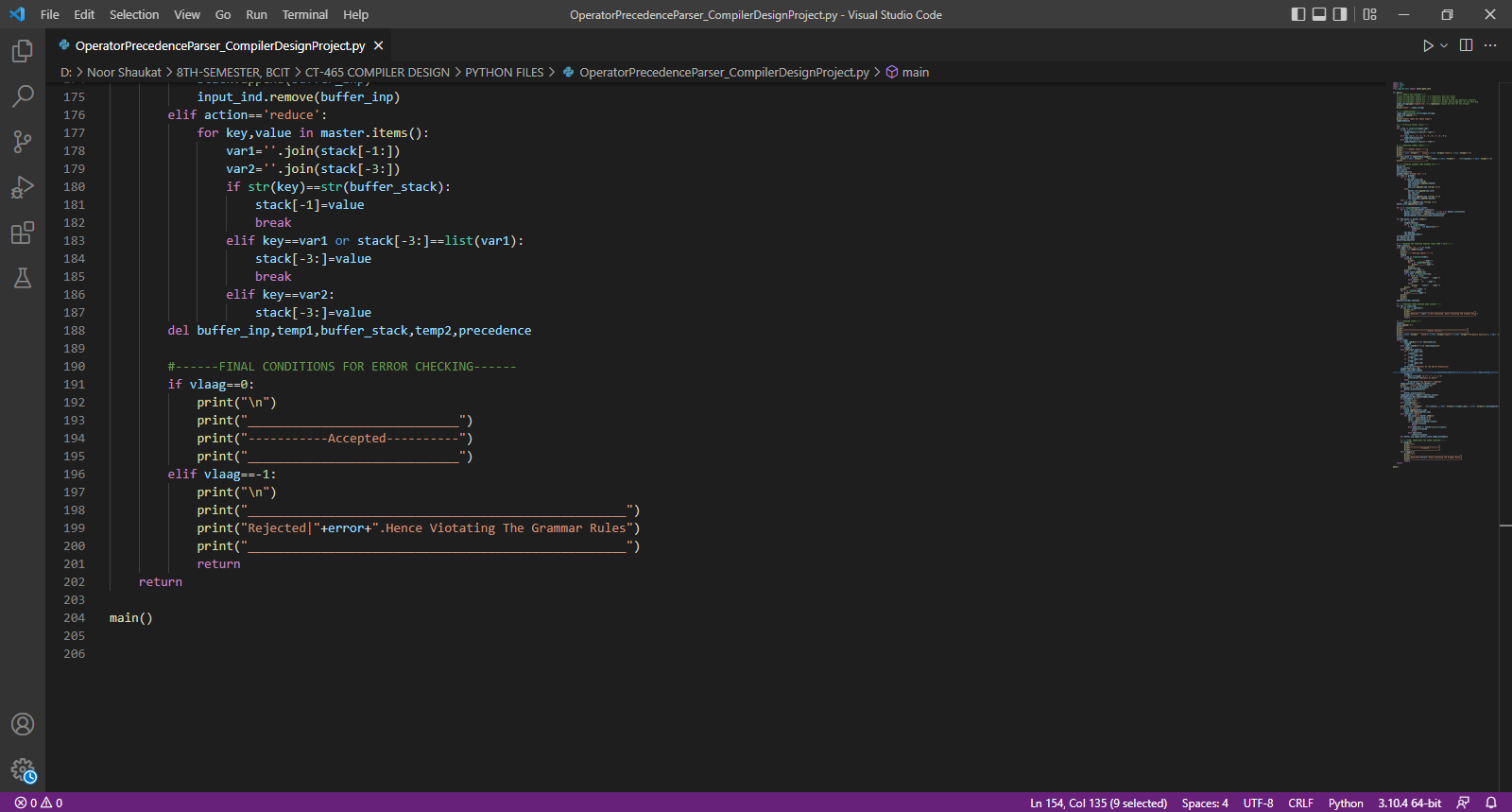










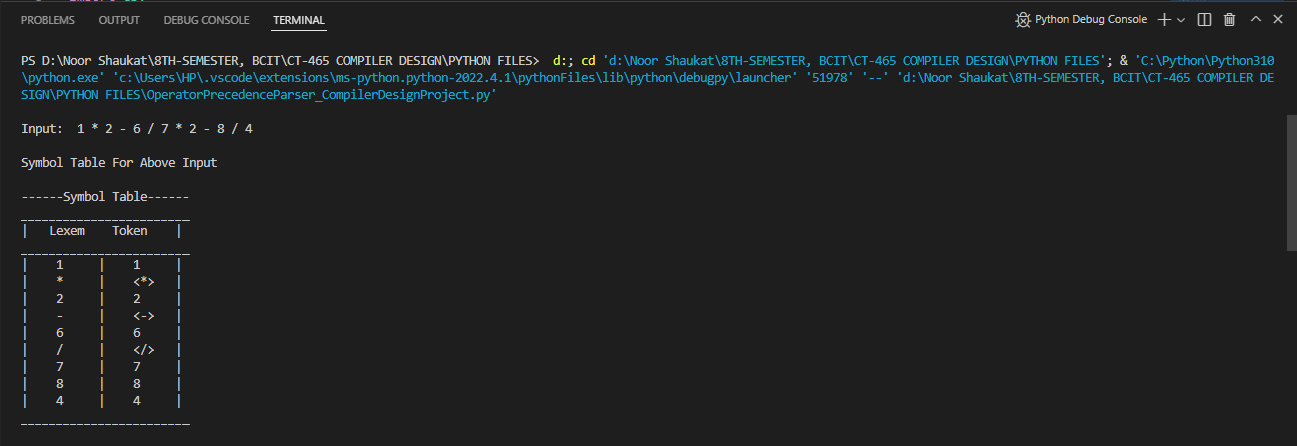


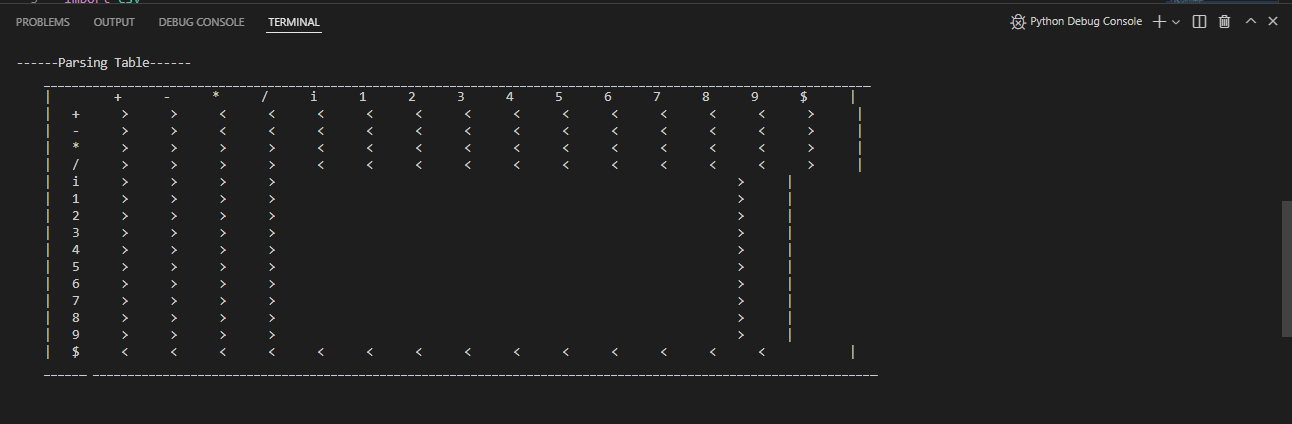
Firstly, multiple inputs are taken from separate number of files, here the inputs are being checked one at a time having the rest of them being commented.

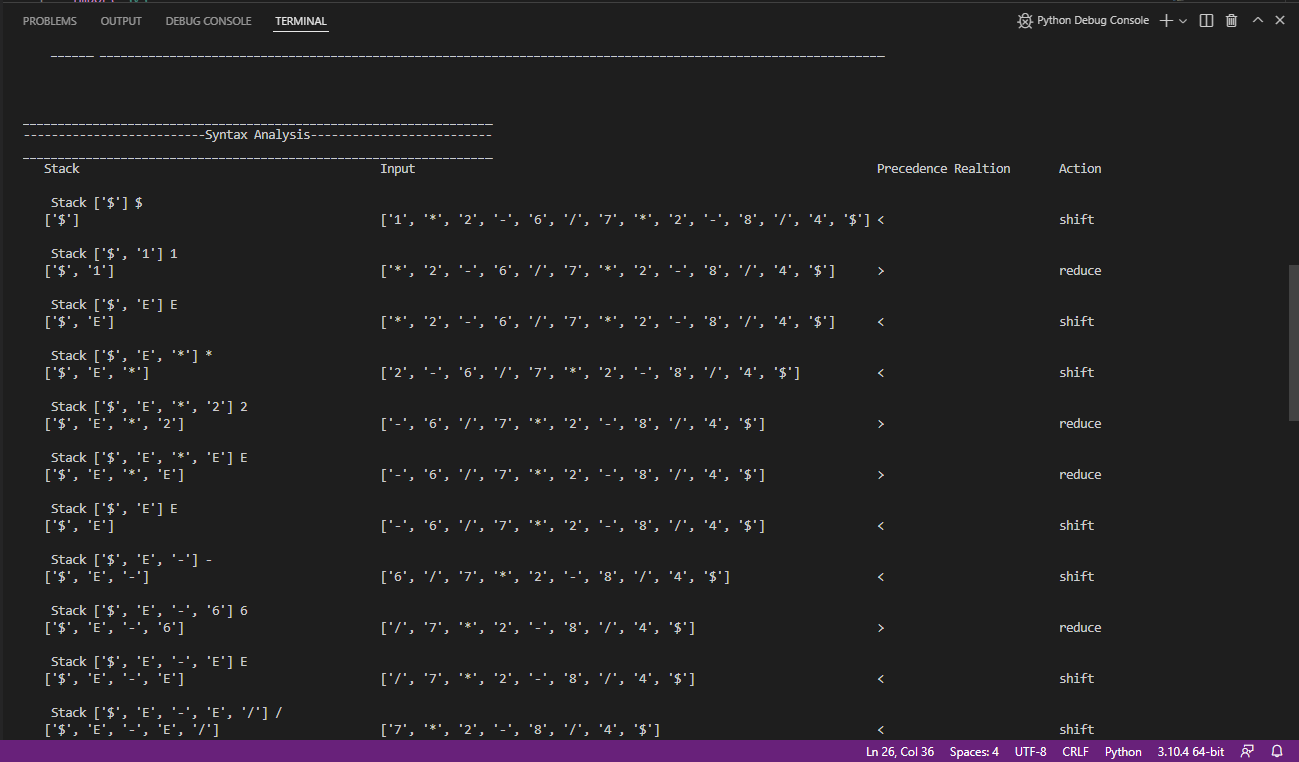
Then as the code executes the input string gets printed initially, then tokenization is being done, next is the symbol table that gets created and printed, the next portion is to read the grammar of the parser from the grammar.txt file, and it’s feed to the a list of new variables in the code, where the grammar rules are marked to be separated by a pipe symbol and at few parts these pipeline symbols are being replaced by arrow symbols written as “->”, after this the parsing table is being read from order.csv which also gets printed containing the rules in the following portion by getting it stored in the order variable, coming to the next portion of the code error handing through printing error messages has been done, proceeding part is the syntax analyzer along with the UI or in other words step by step processing of syntax analyzer to show the clear interpretation of the mechanism and some conditions have been applied as if there is a dollar “$” symbol and the length of the stack is equal to two, that means one is the starting variable and other is the dollar symbol, that states the input is acceptable and make the variable zero, other following conditions are for the error handling where vlaag is converted to minus one “-1”. And the next final portion of the code where if vlaag is zero it’s accepted, an accepted message is being printed else a rejected message is displayed.

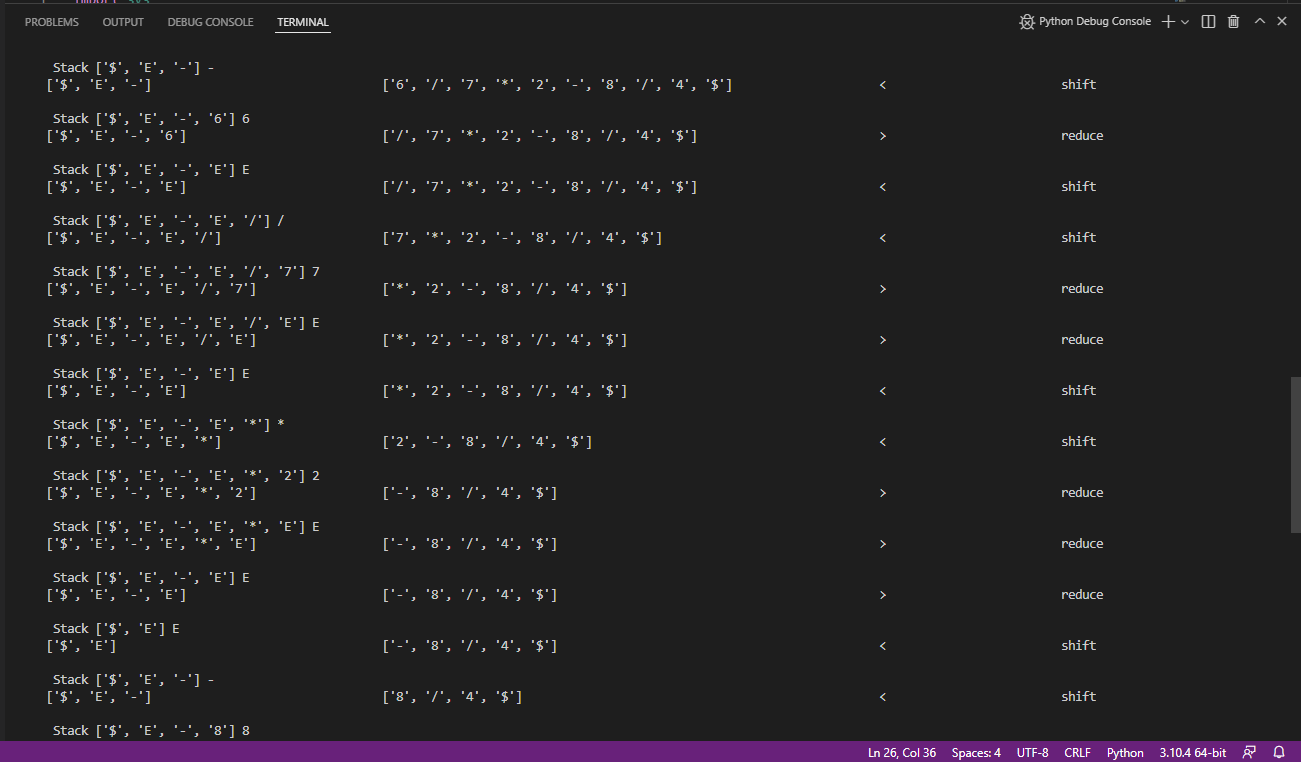
## **TESTING OF INPUT 1**

The syntax analysis table contains three columns stack, input, precedence relation and action. As it’s a bottom-up comparison parsing dollar will be compared to the coming or next symbol that’s one and as we know one has higher precedence according to the principles then that of dollar so that action performed will be shift with the relation of less than, again one is compared to asterisk still one got the higher precedence so the action will be reduce with the relation as greater than. Proceeding the same way to the end where the string input is accepted.





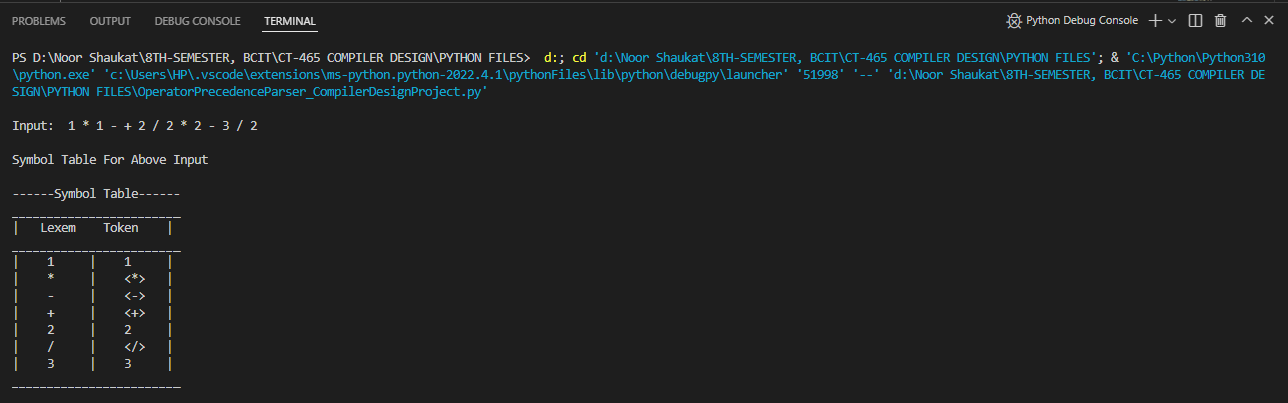


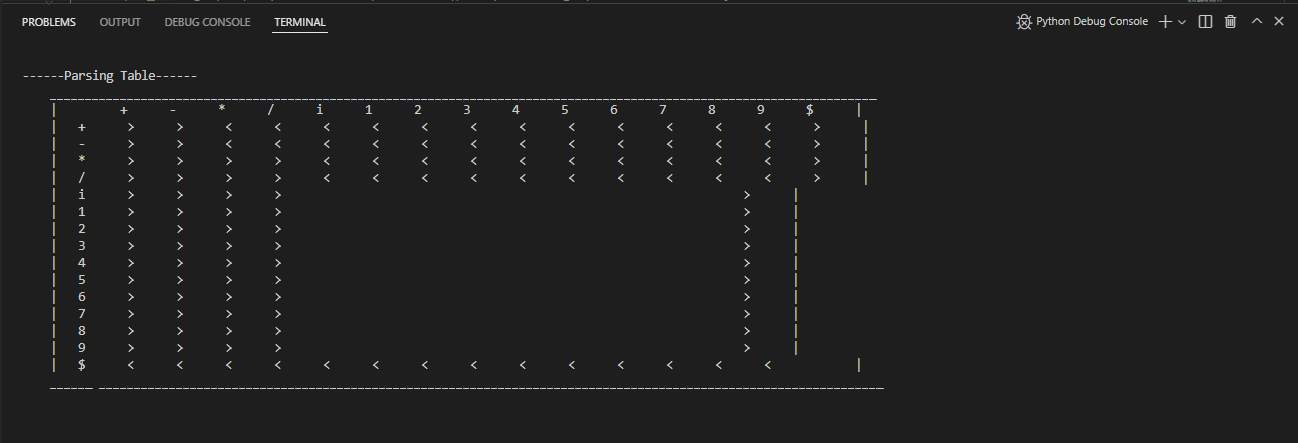


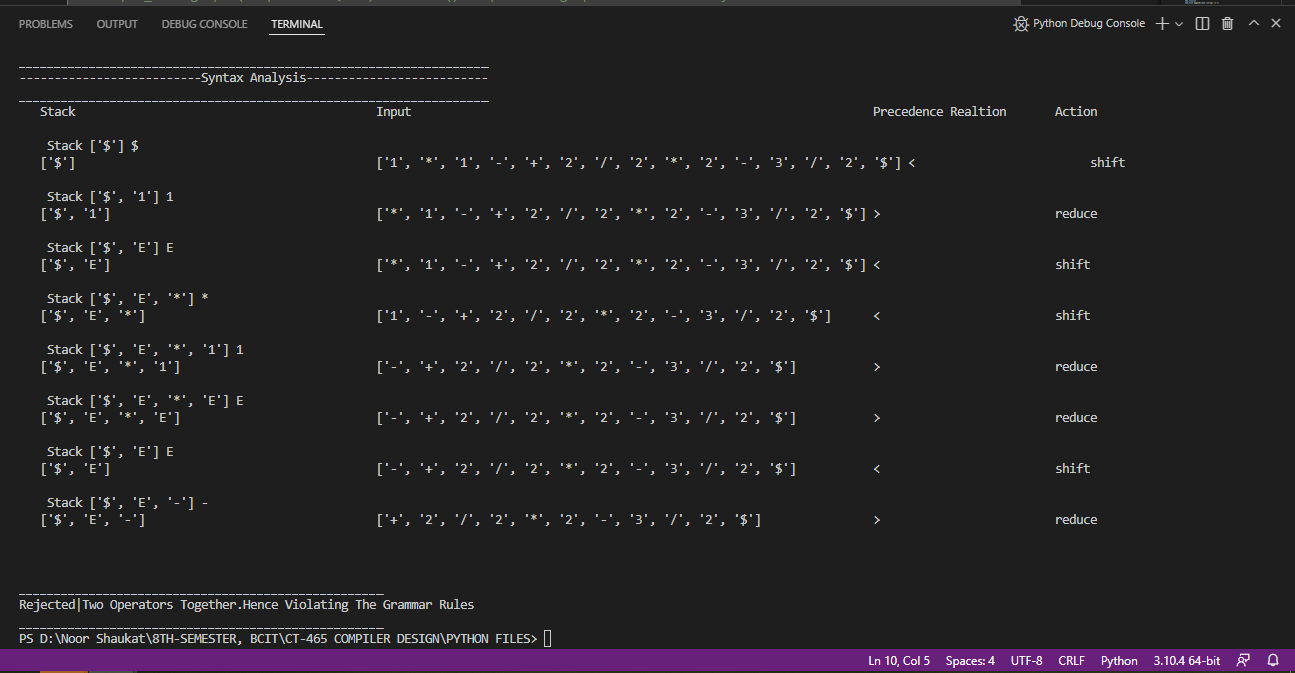


## **TESTING OF INPUT 3**

Here the input gets rejected as it got two operators beside each other, leading to the violation of the defined grammar rules.

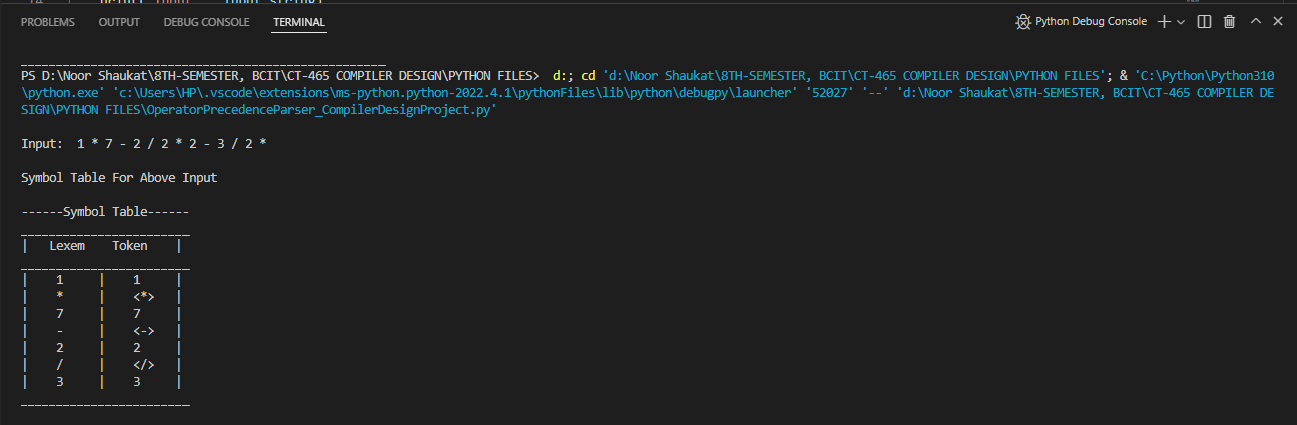
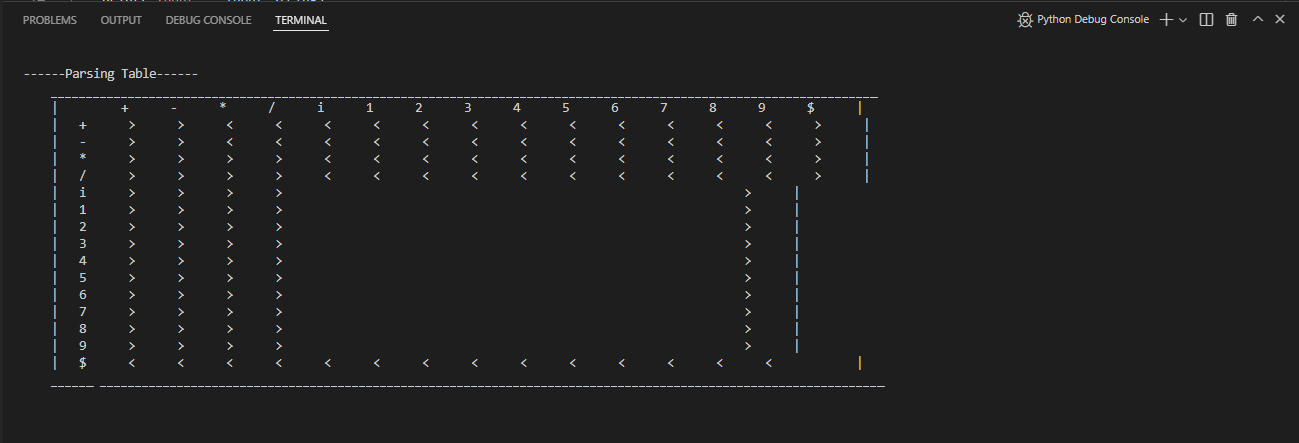


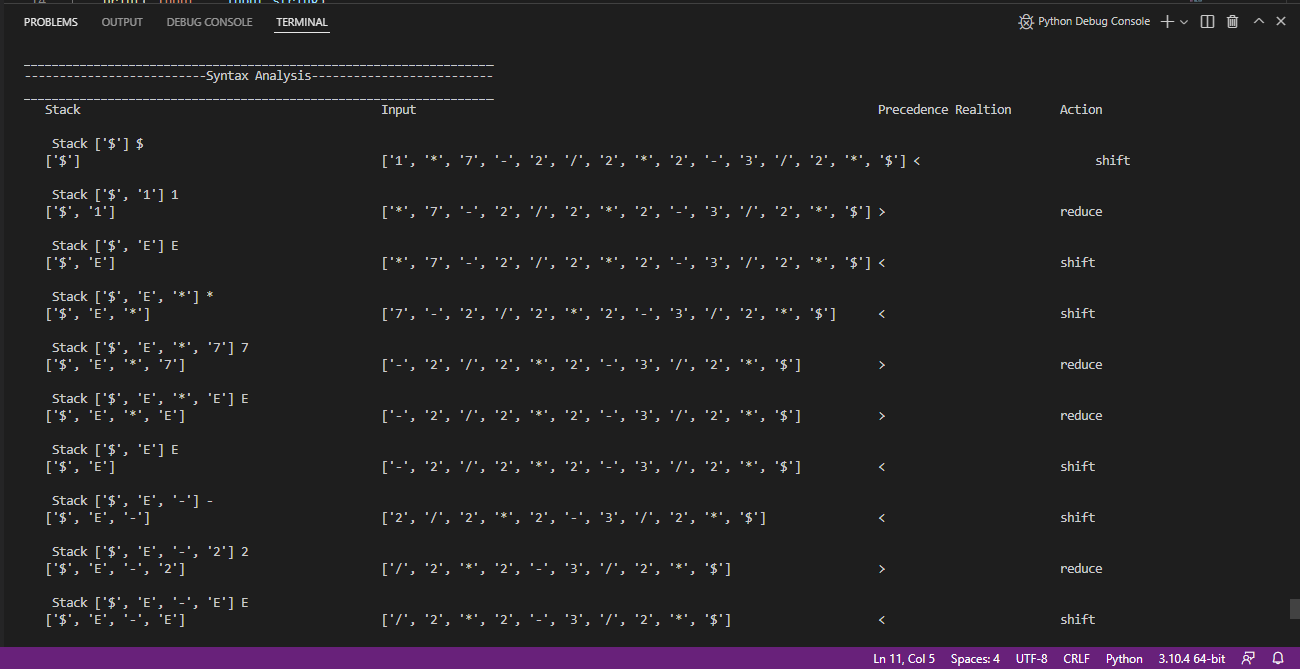


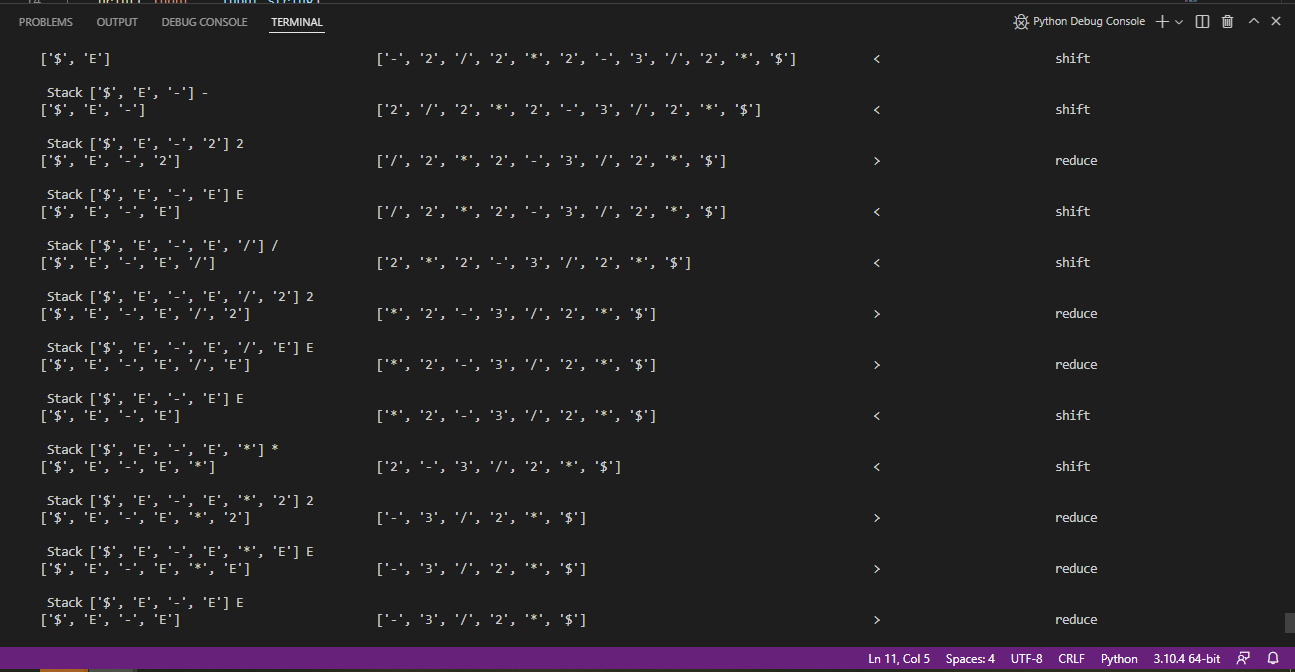


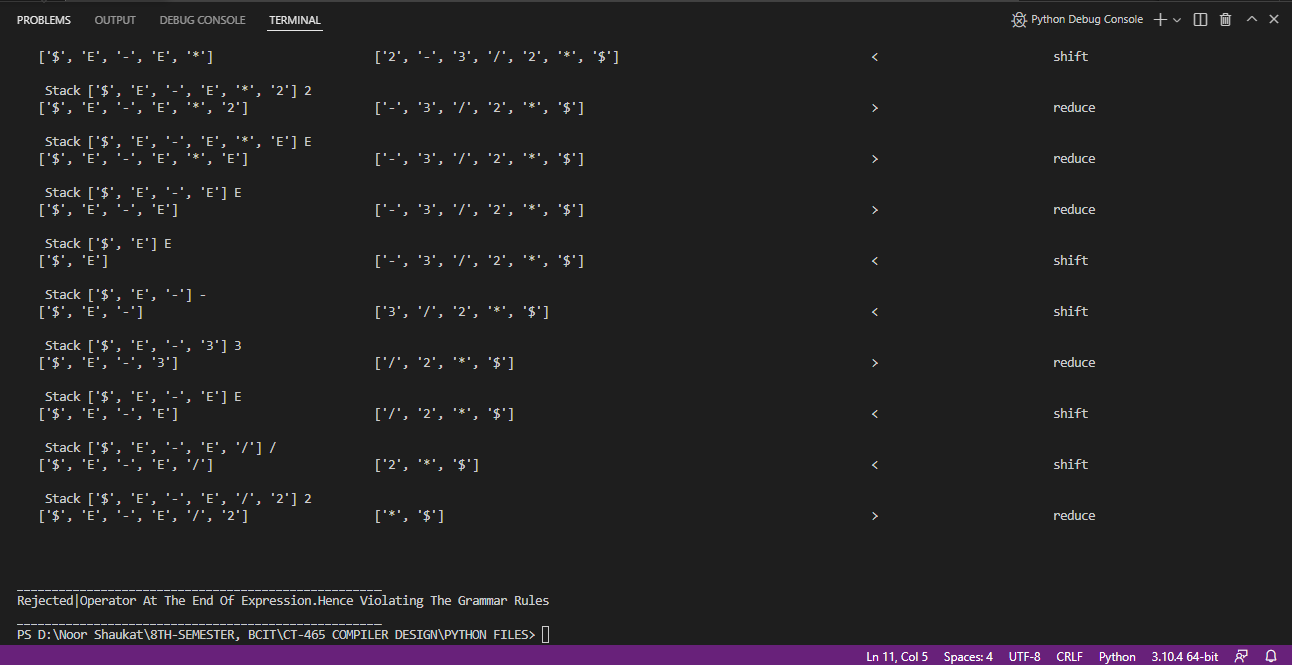
## **TESTING OF INPUT 4**

A rejected input as there’s an operator at the end of the string.







## **TESTING OF INPUT 5**

Rejected as b is an undefined variable to the grammar.

