CS 3342 – Lab 3

Recursive descent parsers in C and Python [100 points]

Due Date: March 12, 2023 (Sunday) - 11:59pm

Part A (in C) (45 points)

1) Copy the program (lab3_recursiveDescentParser.c) in the lab3 folder on Canvas under 'files/labs/lab3'.

This is a recursive descent parser to parse a simple math expression and find the result. For example, it will parse this expression - "2*3+4*5" and find its result (26). To make it simple, only single digit is allowed in the input string in a fixed format for this program.

The parser is coded with the following grammar in BNF:

<expr> -> <term>
| <expr> + <term>
<term> -> <factor>
| <term> * <factor>
<factor> -> digit

Your tasks are:

1. Add logic to check for input error – invalid digit found.
2. Add logic for the following BNF grammar:

<expr> -> <term>
| <expr> + <term>
| <expr> - <term>
<term> -> <factor>
| <term> * <factor>
| <term> / <factor>
<factor> -> digit | (<expr>)

in EBNF grammar:

<expr>	->	<term> { (+ | -) <term>}
<term>	->	<factor> { (* | /) <factor>}
<factor>	->	digit | (<expr>)

3. Change the program to read the input math expression from the terminal.
   i.e., prompt the user: "Please enter a math expression."

Use the following test data – listed inside the program:

```
x = "2*3+4*5";      //test 1 – before change
x = "k2*3+4*5";     //test 2  - for error message
x = "8/2-1*3";      //test 3  - for / and -
x = "8/(4-2)";      //test 4  - for (  )
x = "8*(4-2)+7"     //test 5 – for checking whole logic
```

Pease label your output with the test numbers:

Part A test1:

```
Please enter a math expression.
2*3+4*5
The input expression : 2*3+4*5
Enter Expression : digit is 2
Enter Term : digit is 2
Enter Factor: digit is 2
Enter Factor: digit is 3
factor1 * factor2 : 2 * 3 = 6
Enter Term : digit is 4
Enter Factor: digit is 4
Enter Factor: digit is 5
factor1 * factor2 : 4 * 5 = 20
product1 + product2 : 6 + 20 = 26
**Parsing successful!
result = 26
```

The output for the error message should look like this:

Part A test 2:

```
Please enter a math expression.
k2*3+4*5
The input expression : k2*3+4*5
Enter Expression : digit is k
Enter Term : digit is k
Enter Factor: digit is k
Error: Invalid digit found: k
```

Please put the screen shots of your outputs in a word document. Submit the program file(s) as a single zip-file and the word document separately.

Part B (in Python) (40 points)

1) Copy the program (Lab3_RD_parser_student.py) in the lab3 folder on Canvas under 'files/labs/lab3'.

This is also a recursive descent parser that parses a simple math expression and find the result. The sample Python program can parse expressions with 'plus' operations only. It can read multi-digit numbers though.

Your task is to add code for the following BNF grammar and for the output of the test cases. You can rewrite or write your own Python program for this if you prefer.

```
<expr> -> <term>
        | <expr> + <term>
        | <expr> - <term>
<term> -> <factor>
        | <term> * <factor>
        | <term> / <factor>
<factor> -> digit | (<expr>)
```

in EBNF grammar:

```
<expr>      ->    <term> { (+ | - ) <term>}
<term>      ->    <factor> { (* | / ) <factor>}
<factor>    ->    digit | (<expr>)
```

Use the following input data:

1. 2+3+4+5
2. 20*5 - 4*10
3. 20*5 / 4*5
4. 20*5 / (4*5)

The sample program will give you the following output with the above input data:

```
Please enter a math expression:
2+3+4+5
14.0
Please enter a math expression:
20*5 - 4*10
20.0
(The above answer is wrong.)

Please enter a math expression:
20*5 / 4*5
20
(The above answer is wrong.)
```

Please enter a math expression:
20*5 / (4*5)
Traceback (most recent call last):
……. Errors

You need to fix the logic and add code to have the following output:

Part B test1:
Please enter a math expression:
2+3+4+5
The answer is: 14.0

Part B test2:
Please enter a math expression:
20*5 - 4*10
The answer is: 60.0

Part B test3:
Please enter a math expression:
20*5 / 4*5
The answer is: 125.0

Part B test4:
Please enter a math expression:
20*5 / (4*5)
The answer is: 5.0

Part C (in C++/Java or another language) (15 points)

Rewrite this is a recursive descent parser to parse a simple math expression and find the result in another language (e.g. C++ or Java or..)

Use the same test data in Part A.

Use the following test data

```
x = "2*3+4*5";     //test 1
x = "k2*3+4*5";    //test 2
x = "8/2-1*3";     //test 3
x = "8/(4-2)";     //test 4
```

x = "8*(4-2)+7" //test 5

Please put the screen shots of your test cases in a word document. You can put the screen shots of Part A, Part B, and Part C together in a Word doc. Submit the program file(s) as a single zip-file and the word document separately.