Python Assignment

Q1. Build a mini interpreter that reads a string like "5 + 2 * (3 - 1)" and evaluates it using loops, if–else, and functions.
Code:

```
def precedence(op):       #defining the precedence of operators
    if op == '+' or op == '-':
        return 1
    if op == '*' or op == '/':
        return 2
    return 0


def apply_op(a, b, op):       # how operators should perform
    if op == '+':
        return a + b
    if op == '-':
        return a - b
    if op == '*':
        return a * b
    if op == '/':
        return a // b


def eval_expr(expr):
    values = []          # numbers are stored here from string
    ops = []             # operators are stored here in this
    i = 0
    n = len(expr)
    while i < n:
        if expr[i] == ' ':      #if the expression is empty string then it skips (using continue)
            i += 1
```

```
            continue
        if expr[i].isdigit():         #to check if the taken “i” from string is digit
            val = 0
            while i < n and expr[i].isdigit():
                val = (val * 10) + int(expr[i])     #to get numbers correctly like if we have 25+6 we
#                                                     should get 25 as separate digit.
                i += 1
            values.append(val)
            continue
        if expr[i] == '(':
            ops.append(expr[i])
        elif expr[i] == ')':        #since its stack , we will first face “)”
            while ops and ops[-1] != '(':   #this performs operators until “(“
                op = ops.pop()
                b = values.pop()
                a = values.pop()
                values.append(apply_op(a, b, op))
            ops.pop()          # removes '(' after completion of above operation and continues
        elif expr[i] in '+-*/':    #checks if the operators are +,-,/,*
            while (ops and precedence(ops[-1]) >= precedence(expr[i])):   #checks the precedence
#                                                                           of operators
                op = ops.pop()
                b = values.pop()
                a = values.pop()
                values.append(apply_op(a, b, op))
            ops.append(expr[i])
        i += 1
    while ops:
        op = ops.pop()            #takes out and return the op
        b = values.pop()          #takes out(pop) and returna b value
```

```
        a = values.pop()       #takes out(pops from stack) and returns a value
        values.append(apply_op(a, b, op))
    return values[0]


# Example usage:
expr = "5 + 2 * (3 - 1)"
print(eval_expr(expr))  # Output will be 9
```

Q2. Define a function word_count(sentence) that returns the top N frequent words (case-insensitive, ignore punctuation).

Code:

```
sentence = "Hi Bhanu! oh, let me say your fullname , is it bhanusri? oh yes!"
def word_count(sentence):
    sentence = sentence.lower()  # Make all letters lowercase
    words = sentence.split()     # Splits sentence into words

    word_freq = {}               # Creates empty dictionary to store word counts

    for word in words:
        word = word.strip('.,!?()[]{}"')    # Remove punctuation from the word
        if word == "":
            continue

        if word in word_freq:       # Count the word
            word_freq[word] += 1
        else:
            word_freq[word] = 1

    word_list = list(word_freq.items()) # Convert dictionary to list of (word, count) pairs
    return word_list

result = word_count(sentence)
print(result)
```

Output:

```
[('hi', 1), ('bhanu', 1), ('oh', 2), ('let', 1), ('me', 1), ('say', 1),
('your', 1), ('fullname', 1), ('is', 1), ('it', 1), ('bhanusri', 1),
('yes', 1)]
```

#codechef_compiler