CMPSC-122: Intermediate Programming
Summer 2018

Homework 2

Due Date: 06/22/2018, 11:59PM EST
100 pts

Instructions:

- The work in this assignment must be completed alone.
- The file name must be HW2.py (incorrect name files will get a -10 point deduction)
- When any function returns an error, it must be a string containing "error"
- Do not include test code outside any function in the upload. Remove all your testing code before uploading your file. That includes user-defined input()

Goal:

Modify the function *calculator*(*expr*) so that it supports exponentiation. This operation will be represented in the string as ^ unlike Python's **. In this assignment, more than one consecutive exponentiation is not supported. An example of a valid expression is "-5 + 60 / 3^3 * 4 - 2 * 4^2"
Notes:

- This is a straightforward upgrade of HW1 and there is no starter code
- You will also have to modify findNextOpr and exeOpr to include ^ (which is ** in Python)
- In your submission, include all functions in your HW2.py script (calculator, findNextOpr, isNumber, getNextNumber and exeOpr)

Function requirements:

✓ The function must **return** the computed value if *expr* is a correct formula, otherwise it must return an error message.
✓ When any function returns a numeric value, it must be float
✓ Do not use *exec* or *eval* function. You will not receive credit if your program uses any of the two functions anywhere
✓ All five functions from HW1 must work

Grading Notes:

- The grading script will feed 5 randomly chosen test inputs, each for 20 points. One of them will be an input that should cause an error such as "4 * / 2 + 5 ^", whose expected returned value is an error message.

```
Example:
calculator("-5 + 60 / 3^3 * 4  -  2 * 4^2")
-28.11111111111111
>>> calculator("4^2 /  2^2")
4.0
>>> calculator("-4^ /  2^2")
'error'
```

Deliverables:

- Include all the functions in your script named HW2.py. Submit it to the HW2 CANVAS assignment before the due date

Tips for incomplete HW1:

Assume you call calculator on the expression “2*3-5-4/2*3+6”, thus:

expr	2	*	3	-	5	-	4	/	2	*	3	+	6
pos	0	1	2	3	4	5	6	7	8	9	10	11	12

while True:

when pos = 6

2	*	3	-	5	-	4	/	2	*	3	+	6
0	1	2	3	4	5	6	7	8	9	10	11	12

mode=”add”
Save:
add_result= 2*3-5 =1
addmode_lastOpr=”-”

mode=”mul”
Start:
mul_result=4

This means that the analysis of the expression would look like:

```
elif (newOpr=="*" or newOpr=="/") and mode=="add":
      mul_result = newNumber
      add_lastOpr = opr
      mode="mul"
```

So when pos=10, you should finish the multiplication mode computing mul_result=4/2*3, which is 6, and move back to the addition mode. You can achieve this by using the saved add_result=1 and add_lastOpr=”-“ like this:

```
elif (newOpr=="+" or newOpr=="-") and mode=="mul":
      mul_result = exeOpr(mul_result, opr, newNumber)
      add_result = exeOpr(add_result, addmode_lastOpr, mul_result)
      mode="add"
```