CS206A Data Structure

HW1

studentID: 20160253

name: 박예기 (Park ye gi)

Problem1. The below table shows the intermediate steps of converting to a postfix expression.

Token	Stack				Output
	[0]	[1]	[2]	[3]	
3					3
+	+				3
X	+				3 X
*	+	*			3 X
(	+	*	(		3 X
Y	+	*	(		3 X Y
-	+	*	(	-	3 X Y
12	+	*	(	-	3 X Y 12
)	+	*			3 X Y 12 –
-	-				3 X Y 12 - * +
Z	-				3 X Y 12 - * + Z
eos					3 X Y 12 - * + Z -

Therefore, the postfix expression for "3+X*(Y-12)-Z" is **"3 X Y 12 - * + Z –"**.

Problem 2(a)

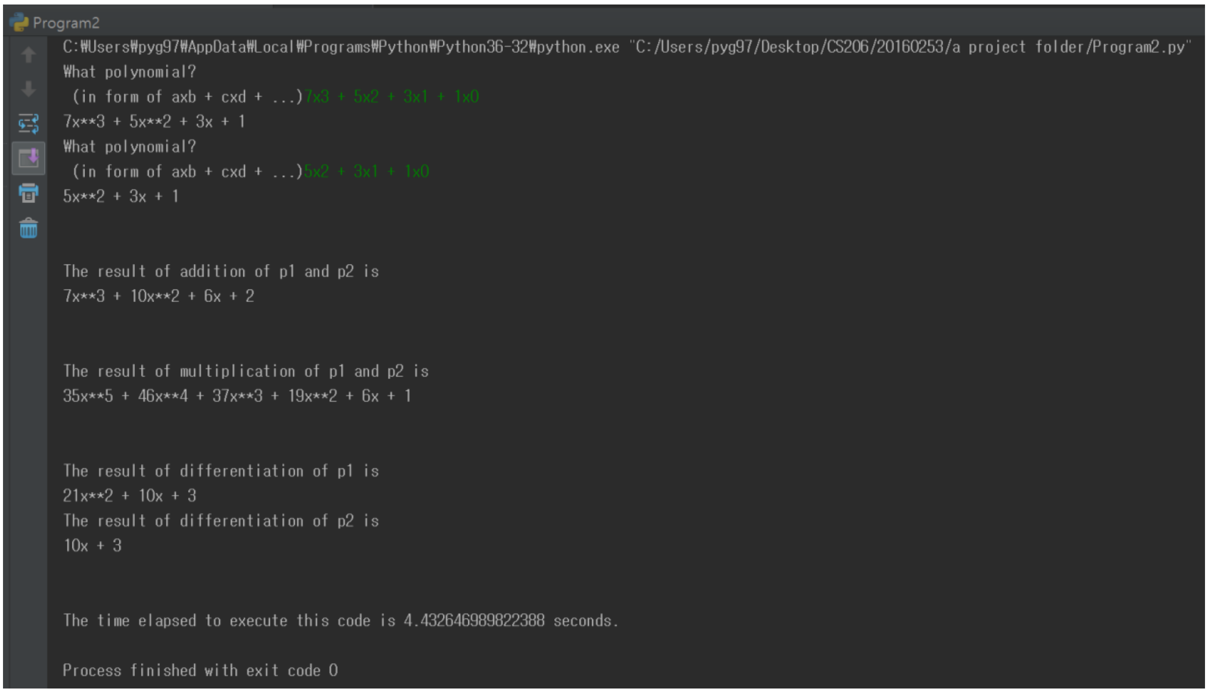
```
Run  Program1
C:₩Users₩pyg97₩AppData₩Local₩Programs₩Python₩Python36-32₩python.exe C:/Users/pyg97/PycharmProjects/DS_HW1/Program1.py
What polynomial?
 (in form of axb + cxd + ...)5x2 + 3x1 + 1x0
5x**2 + 3x + 1
The time elapsed to execute this code is 1.4637048244476318 seconds.

Process finished with exit code 0
```

Let input format axb + cxd + … (e.g. 5x2 + 3x1 + 1x0). Then, the Program1 produces ax**b + cx**d +… (e.g. 5x**2 + 3x + 1)

I used class to make object type Polynomial. This class Polynomial has construct function(def __init__()). The construct function gets itself, coefficient, and degree for parameters, and in this case coefficient and degree are lists that represent coefficients and exponents each. The construct function prints the output formula. It firstly seperates whether the clause is the first or not and then seperates that exponents is 0, exponent is 1, and the other case. By connecting string, it finally makes an output formula as a string.

I made a function that creates polynomial object. This function splits the input string according to ' + ' and 'x' . Next, the function seperates the coefficient number and the exponent number. Then it makes two lists, for coefficient and exponent each, to use them for making Polynomial object.

Problem2(b)



```
Program2
C:₩Users₩pyg97₩AppData₩Local₩Programs₩Python₩Python36-32₩python.exe "C:/Users/pyg97/Desktop/CS206/20160253/a project folder/Program2.py"
What polynomial?
 (in form of axb + cxd + ...)7x3 + 5x2 + 3x1 + 1x0
7x**3 + 5x**2 + 3x + 1
What polynomial?
 (in form of axb + cxd + ...)5x2 + 3x1 + 1x0
5x**2 + 3x + 1


The result of addition of p1 and p2 is
7x**3 + 10x**2 + 6x + 2


The result of multiplication of p1 and p2 is
35x**5 + 46x**4 + 37x**3 + 19x**2 + 6x + 1


The result of differentiation of p1 is
21x**2 + 10x + 3
The result of differentiation of p2 is
10x + 3


The time elapsed to execute this code is 4.432646989822388 seconds.

Process finished with exit code 0
```

Let input format axb + cxd + ... (e.g. 5x2 + 3x1 + 1x0). Then, the Program2 produces ax**b + cx**d +... (e.g. 5x**2 + 3x + 1)

The different points with Program1(code for problem 2(a)) is Program2 has functions inside class Polynomial so that these functions represent operations : addition, multiplication, and differentiation.

The addition operation makes 2 lists for coefficients and exponents of addition polynomial in it. Then, the function checks whether there is the clause that has a certain degree from the maximum degree of 2 polynomials. If the degree is in both 2 polynomials, the addition value is appended to list. If the degree is in only one polynomial, the original value is appended to list.

The multiplication operation also makes 2 lists in it, like addition operation. The function uses two "**for** expression" to multiply each clause of two polynomials. In multiplication of one clause in the first polynomial and one clause in the second polynomial, if the new exponent already exists in exponent list, the function adds the coefficients that have the same exponent value. In the other case, the function just appends new exponential and new coefficient to each list.

The differentiation operation also makes 2 lists in it. From the maximum exponential clause, the function multiplies the exponent and the coefficient, and it reduces the exponential -1.

Problem2(c)

ADT *Polynomial Calculus* is

objects : $p(x) = c_1x^{e1} + c_2x^{e2} + \ldots + c_nx^{en}$; sum of clauses that have coefficient c_i and exponent e_i, e_i are integers >=0

functions :

for all *poly, poly1, poly2* ∈ Polynomial

Polynomial Addition(poly1, poly2)	::=	**return** the polynomial poly1 + poly2
Polynomial Multiplication(poly1, poly2)	::=	**return** the polynomial poly1 x poly2
Polynomial Differentiation(poly)	::=	**return** a derivative (poly)'

end *Polynomial Calculus*

Problem2(d)

Quotient remainder theorem can be added to the Polynomial Calculus ADT. If user inputs two polynomials, then this operation gives outputs which are quotient and remainder when the first input polynomial is divided by the second input polynomial.

Also, as a converse of differentiation, the indefinite integral can be added to the Polynomial Calculus ADT. If user inputs one polynomial, then this operation gives an output polynomial which is the indefinite polynomial of the input polynomial.