PART A

```
import re
def isValidAE(s):
        #remove all whitespaces
        s = ''.join(s.split())
        #go through the string to see if there are any non-digit char that's not +
or *
        if bool(re.search('Wd',s)) == True:
            if bool(re.search('WD',s)) == True:
                return bool(re.search('W*|W+', s))
        return True

def add(s):
    #split on +
    addlist = s.split("+")
    totalA = 0
    for i in range(len(addlist)):
        totalA += mult(addlist[i])
    return totalA

def mult(s):
    #split the string
    multlist = s.split("*")
    #edge case for if there's no *
    if len(multlist) == 1:
        return int(s)
    else:
        totalM = 1
        for i in range(len(multlist)):
            totalM *= int(multlist[i])
        return totalM

def errorM():
    return "illegal Expression"

def evalAE(s):
    #if the input is end, terminate
    if isValidAE(s) == True:
        s = ''.join(s.split())
        return add(s)
    #if invalid
    else:
        return errorM()
```

TEST A

```
C:₩Python27₩python.exe C:/Users/Ig/Desktop/Hw4.ArithExpPositorn/testHw4PartA.py
::::: Test of Hw4 Part A :::::::

TEST of isValidAE -------
PASS
PASS
PASS
PASS
PASS
PASS
PASS

TEST of evalArithmeticExpr -------
PASS
PASS
PASS
PASS
PASS

Process finished with exit code 0
```

REPL A

```
C:₩Python27₩python.exe C:/Users/Ig/Desktop/Hw4.ArithExpPositorn/hw4PartARepl.py
REPL: Enter Expression with + and * . Type 'end' to quit
>>>1+1
2
>>>1+1+1
2
>>>1/1
illegal Expression
>>>end

Process finished with exit code 0
```

Part B

```
import re

def add(s):
    #split on +
    addlist = s.split("+")
    totalA = 0
    for i in range(len(addlist)):
        totalA += mult(addlist[i])
    return totalA

def mult(s):
    #split the string
    multlist = s.split("*")
    #edge case for if there's no *
    if len(multlist) == 1:
        return int(s)
    else:
        totalM = 1
        for i in range(len(multlist)):
            totalM *= int(multlist[i])
        return totalM

def addVar(s):
    #split on +
    addlist = s.split("+")
    totalA = 0
    for i in range(len(addlist)):
        totalA += multVar(addlist[i])
    return totalA

def multVar(s):
    multlist = s.plit("*")
    if len(multlist) == 1:
        #if the char is letter
        if re.search('[a-zA-Z]',s):
            if s in defined:
                return int(mem.get(s))
            else:
                return errorM()
        #if it is a number
        else:
```

```
            return int(s)
        #more than one var, splited by *
    else:
        totalM = 1
        for i in range(len(multlist)):
            if re.search('[a-zA-Z]]',multlist[i]):
                if multlist[i] in defined:
                    totalM *= int(mem.get(multlist[i]))
                else:
                    return errorM()
            else:
                totalM *= int(multlist[i])
        return totalM

def errorM():
    return "illegal Expression"

def isValidA(s):
    if bool(re.search('Wd', s)) == True:
        if bool(re.search('WD', s)) == True:
            return bool(re.search('W*|W+', s))
    return True

def assignSingle(s):
    s = ''.join(s.split())
    assign = s.split('=')
    if len(assign) > 2:
        return errorM()
    else:
        varName = assign[0]
        expr = assign[1]
        assignVal = add(expr)
        mem[varName] = assignVal
        defined.append(varName)
        return varName

def assignMult(s):
    s= ''.join(s.split())
    assign = s.split('=')
    if len(assign) > 2:
        return errorM()
    else:
        varName = assign[0]
        expr = assign[1]
        assignVal = addVar(expr)
        mem[varName] = assignVal
        defined.append(varName)
```

```
        return varName

def checkDefined(s):
    s = ''.join(s.split())
    if s in defined:
        return True
    else:
        return False

def isValidPositron(s):
    s = ''.join(s.split())

    #if Wd[a-zA-Z]
    if re.search('Wd[a-zA-Z]',s) or re.search('Ww(-|/)',s):
        return False
    elif bool(re.search('Wd+=Wd+',s)) and not bool(re.search('[a-zA-Z]',s)):
        return False
    #arithmetic
    elif re.search('Wd+',s):
        return True
    elif re.search('Ww+(W+|W*)Ww+',s):
        return True
    #single num assign
    elif re.search('[a-zA-Z]=Wd+',s):
        return True
    #var assign expression (var)
    elif re.search('[a-zA-Z]=Ww+(W+|W*)Ww+',s):
        return True
    #var +* var
    elif re.search('[a-zA-Z](W+|W*)[a-zA-Z]',s):
        return True
    #single var
    elif re.search('[a-zA-Z]',s):
        return True
    else:
        return False

defined = []
mem = {}   # you need a data structure to keep track of your variables in memory
def evalPositronStmt(s):

    if isValidPositron(s):
        s = ''.join(s.split())

        # if Alphabet
        if any(c.isalpha() for c in s):
            # var = dig
```

```
            # expression with defined Var
            if re.search('[a-zA-Z]Ww+=[a-zA-Z]Ww+(W+|W*)Wd+',s):
                return mem.get(assignMult(s))
            elif re.search('[a-zA-Z]Ww+(W+|W*)Wd+',s):
                return addVar(s)
            elif re.search('[a-zA-Z]=Wd+', s):
                return mem.get(assignSingle(s))
            # defined Var
            elif re.search('[a-zA-Z]', s):
                if s in defined:
                    return int(mem.get(s))

        # #if no Alphabet
        elif re.search('Wd+(W+|W*)Wd+', s):
            return add(s)
        elif re.search('Wd+', s):
            return int(s)
    else:
        return errorM()
```

TEST B

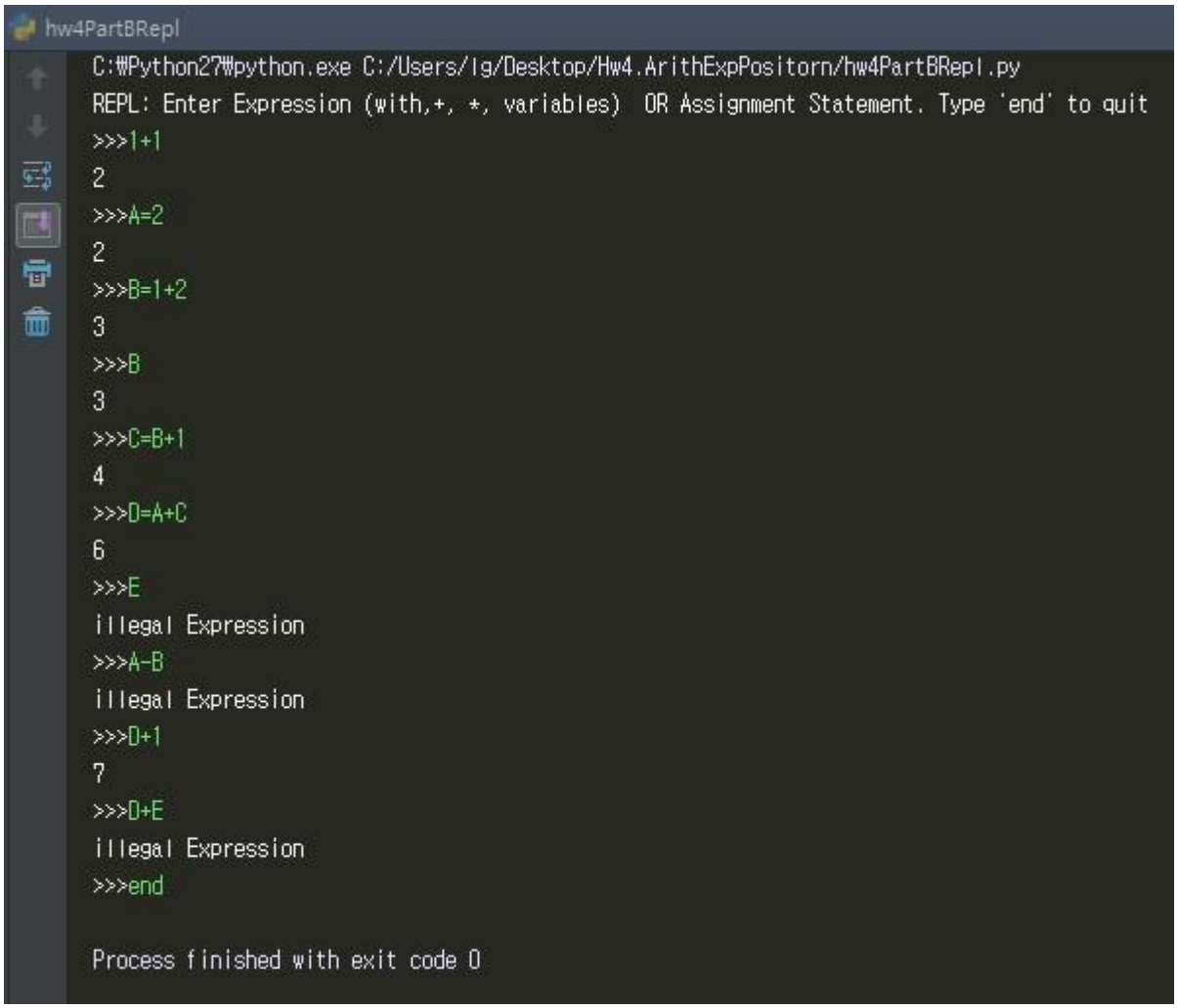
```
testHw4PartB
C:₩Python27₩python.exe C:/Users/Ig/Desktop/Hw4.ArithExpPositorn/testHw4PartB.py
:::::::: Testing HW4 Part B  ::::::::::::::::

TEST of isValid Positron Sentence-------
PASS
PASS
PASS
PASS
PASS
PASS
PASS

TEST of evalPositron Sentence -------
PASS
PASS
PASS
PASS
PASS

Process finished with exit code 0
```

REPL B


hw4PartBRepl
C:₩Python27₩python.exe C:/Users/lg/Desktop/Hw4.ArithExpPositorn/hw4PartBRepl.py
REPL: Enter Expression (with,+, *, variables) OR Assignment Statement. Type 'end' to quit
>>>1+1
2
>>>A=2
2
>>>B=1+2
3
>>>B
3
>>>C=B+1
4
>>>D=A+C
6
>>>E
illegal Expression
>>>A-B
illegal Expression
>>>D+1
7
>>>D+E
illegal Expression
>>>end
Process finished with exit code 0