Experiment - 8 WAP to generate 3 address code

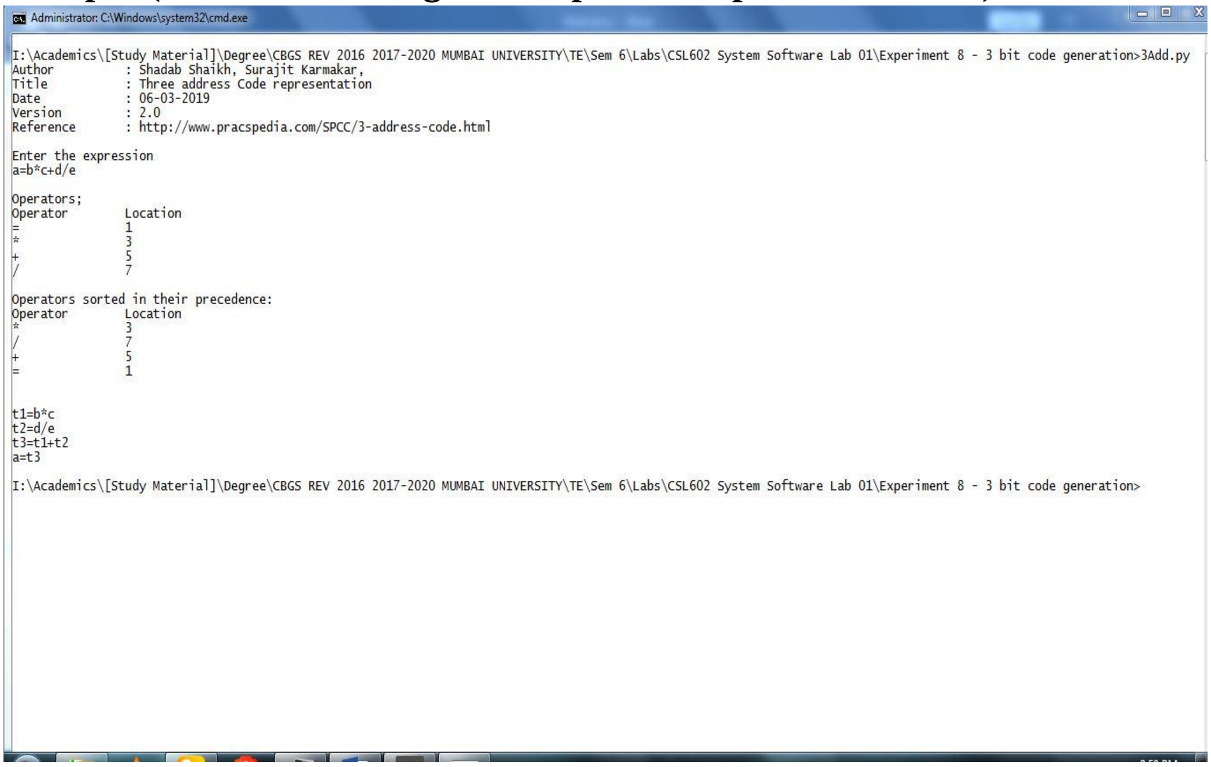
Name: Shaikh Shadab **Rollno : 17DCO74**
Class : TE.CO **Batch : B3**

#Source Code in python

```
__author__ = 'Shadab Shaikh, Surajit Karmakar,'
__title__ = 'Three address Code representation'
__date__ = '06-03-2019'
__version__ = '2.0'
__link__ = 'http://www.pracspedia.com/SPCC/3-address-code.html'#Reference author link
print('Author           : ' + __author__)
print('Title            : ' + __title__)
print('Date             : ' + __date__)
print('Version          : ' + __version__)
print('Reference        : ' + __link__)
precedence=[['/','1'],['*','1'],['+','2'],['-','2'],['^','0'],['=','3']]
#creating a matrix of precedence lowest number highest precedence
def precedenceOf(t):                            #checking character precedence
        token=t[0]                              #assigning string to 1 character variable
        for i in range(len(precedence)):
                if(token==precedence[i][0]):
                        #checking if character matches precedence matrix
                        return int(precedence[i][1]+"")      #returning its precedence value
        return -1                                            #or returning false
opc=0                                   #initialization of opc
token=''                                #acting as a pointer to character
operators=[[],[],[],[],[],[],[],[],[],[],[],[]]      #creating 10*2 space for operator
expr=""                                 #will store the user input
temp=""                                 #used for soring
expr=input("\nEnter the expression\n")
processed=[]                            #using to see if literal is already processed
for i in range(len(expr)):
        processed.append(False)         #initialization of process mat with false
for i in range(len(expr)):
        token=expr[i]                   #scanning each character in expr mat
        for j in range(len(precedence)):
                if(token==precedence[j][0]):  #if char matches with precedence mat character
                        operators[opc].append(token+"")
                        operators[opc].append(str(i)+"")#appending it to operator matrix
                        opc+=1                  #incrementing opc for further storing
                        break

print("\nOperators;\nOperator\tLocation")
for i in range(opc):
        print(operators[i][0]+"\t\t"+operators[i][1])
        #printing operator found and their location
for i in range(opc-1,0,-1):#sorting matrix descending based on precedence level of operator
        for j in range(i):
                if(precedenceOf(operators[j][0]) > precedenceOf(operators[j+1][0])):
                        temp=operators[j][0]
                        operators[j][0]=operators[j+1][0]
                        operators[j+1][0]=temp
                        temp=operators[j][1]
                        operators[j][1]=operators[j+1][1]
                        operators[j+1][1]=temp

print("\nOperators sorted in their precedence:\nOperator\tLocation")
for i in range(opc):
        print(operators[i][0]+"\t\t"+operators[i][1])           #displaying sorted result
print("\n")
for i in range(opc):                                    #running for loop with operator count range
        j=int(operators[i][1]+"")               #stores the number of precedence value
        op1="
        op2="                                           #will be storing operand 1 and 2
        if(processed[j-1]==True):               #determining if literal is already processed
                if(precedenceOf(operators[i-1][0])==precedenceOf(operators[i][0])):
                        op1="t"+str(i)  #if precedence matches making t# as new operand
                else:
                        for x in range(opc):
                                if((j-2)==int(operators[x][1])):

                                        op1="t"+str((x+1))+""
                                        #making left most t# operand, the middle t# operand
        else:
                op1=expr[j-1]+""        #else making middle character 1st operand
        if(processed[j+1]==True):       #checking if rightmost is already processed
                for x in range(opc):
                        if((j+2)==int(operators[x][1])):
                                op2="t"+str((x+1))+""
                                        #making right most t# operand, the middle t# operand
        else:
                op2=expr[j+1]+""        #else making right most character 2nt operand
        if(operators[i][0]=='='):                       #checking if operator matches equal operator
                op2="t"+str((x))+""                     #using the latest t# variable
                print(op1+operators[i][0]+op2)#printing only operand with operator
                processed[j]=processed[j-1]=processed[j+1]=True
                                                        #updating processed matrix
        else:
                print("t"+str((i+1))+"="+op1+operators[i][0]+op2)
                                        #printing t# with = operand 1 , operator and operand 2
```

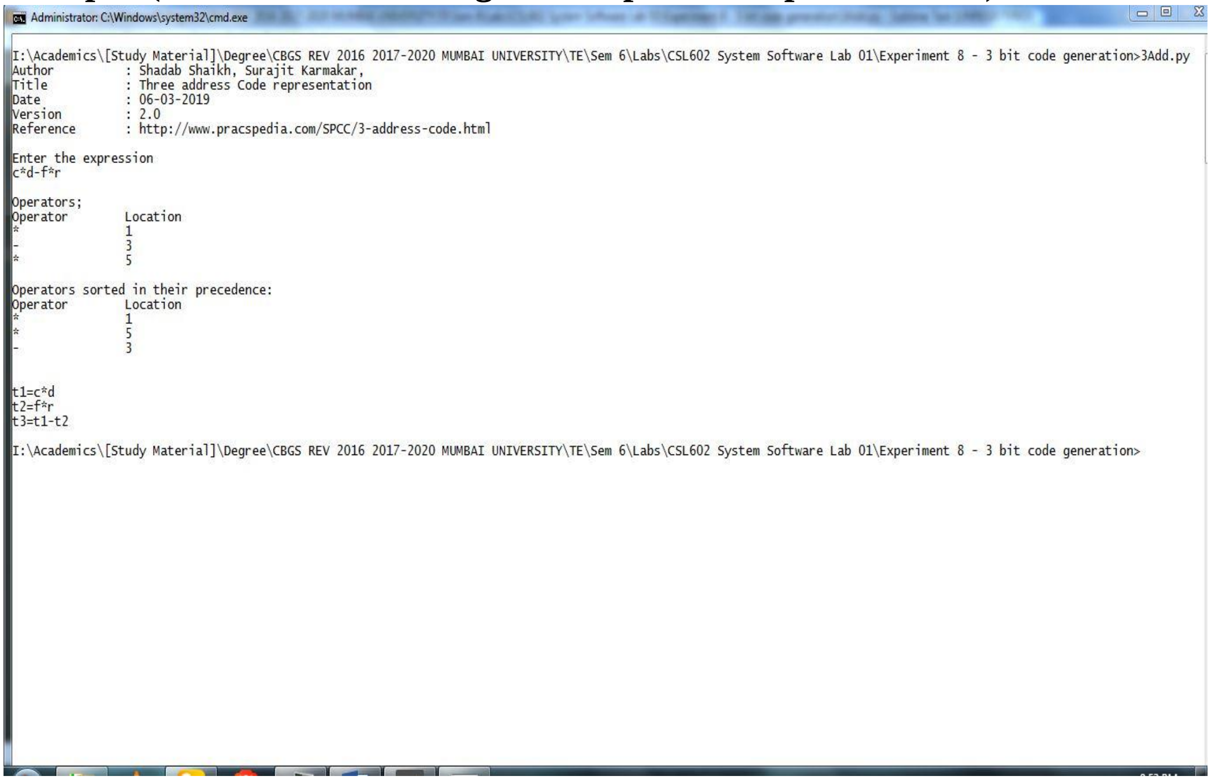
```
                    processed[j]=processed[j-1]=processed[j+1]=True
                                                                  #updating processed matrix
```

#Output (Case 1 With assignment operator input : a=b*c+d/e)



```
I:\Academics\[Study Material]\Degree\CBGS REV 2016 2017-2020 MUMBAI UNIVERSITY\TE\Sem 6\Labs\CSL602 System Software Lab 01\Experiment 8 - 3 bit code generation>3Add.py
Author          : Shadab Shaikh, Surajit Karmakar,
Title           : Three address Code representation
Date            : 06-03-2019
Version         : 2.0
Reference       : http://www.pracspedia.com/SPCC/3-address-code.html

Enter the expression
a=b*c+d/e

Operators;
Operator        Location
=               1
*               3
+               5
/               7

Operators sorted in their precedence:
Operator        Location
*               3
/               7
+               5
=               1


t1=b*c
t2=d/e
t3=t1+t2
a=t3

I:\Academics\[Study Material]\Degree\CBGS REV 2016 2017-2020 MUMBAI UNIVERSITY\TE\Sem 6\Labs\CSL602 System Software Lab 01\Experiment 8 - 3 bit code generation>
```

#Output (Case 2 Without assignment operator input : c*d-f*r)



```
I:\Academics\[Study Material]\Degree\CBGS REV 2016 2017-2020 MUMBAI UNIVERSITY\TE\Sem 6\Labs\CSL602 System Software Lab 01\Experiment 8 - 3 bit code generation>3Add.py
Author          : Shadab Shaikh, Surajit Karmakar,
Title           : Three address Code representation
Date            : 06-03-2019
Version         : 2.0
Reference       : http://www.pracspedia.com/SPCC/3-address-code.html

Enter the expression
c*d-f*r

Operators;
Operator        Location
*               1
-               3
*               5

Operators sorted in their precedence:
Operator        Location
*               1
*               5
-               3


t1=c*d
t2=f*r
t3=t1-t2

I:\Academics\[Study Material]\Degree\CBGS REV 2016 2017-2020 MUMBAI UNIVERSITY\TE\Sem 6\Labs\CSL602 System Software Lab 01\Experiment 8 - 3 bit code generation>
```