**THREE ADDRESS CODE GENERATION QUADRUPLE,TRIPLE**

Code:

OPERATORS = set(['+', '-', '\*', '/', '(', ')'])

PRI = {'+':1, '-':1, '\*':2, '/':2}

### INFIX ===> POSTFIX ###

def infix\_to\_postfix(formula):

    stack = [] # only pop when the coming op has priority

    output = ''

    for ch in formula:

        if ch not in OPERATORS:

            output += ch

        elif ch == '(':

            stack.append('(')

        elif ch == ')':

            while stack and stack[-1] != '(':

                output += stack.pop()

            stack.pop() # pop '('

        else:

            while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:

                output += stack.pop()

            stack.append(ch)

    # leftover

    while stack:

        output += stack.pop()

    return output

def T\_A\_C(exp):

    exp\_stack = []

    t = 1

    pos = infix\_to\_postfix(exp)

    for i in pos:

        if i not in OPERATORS:

            exp\_stack.append(i)

        else:

            print(f't{t} := {exp\_stack[-2]} {i} {exp\_stack[-1]}')

            exp\_stack=exp\_stack[:-2]

            exp\_stack.append(f't{t}')

            t+=1

def Quadruple(exp):

        stack = []

        op = []

        x = 1

        postfix = infix\_to\_postfix(exp)

        print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format('op','arg1','arg2','result'))

        for i in postfix:

            if i in "abcdefghijklmnopqrstuvwxyz" or i in "0123456789":

                stack.append(i)

            elif i == '-':

                op1 = stack.pop()

                stack.append("t%s" %x)

                print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i,op1,"(-)"," t(%s)" %x))

                x = x+1

                if stack != []:

                    op2 = stack.pop()

                    op1 = stack.pop()

                    print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format("+",op1,op2," t(%s)" %x))

                    stack.append("t%s" %x)

                    x = x+1

            elif i == '=':

                    op2 = stack.pop()

                    op1 = stack.pop()

                    print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i,op2,"(-)",op1))

            else:

                op1 = stack.pop()

                op2 = stack.pop()

                print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i,op2,op1," t%s" %x))

                stack.append("t%s" %x)

                x = x+1

def Triple(exp):

        stack = []

        op = []

        x = 0

        postfix =infix\_to\_postfix(exp)

        print("{0:^4s} | {1:^4s} | {2:^4s}".format('op','arg1','arg2'))

        for i in postfix:

            if i in "abcdefghijklmnopqrstuvwxyz" or i in "0123456789":

                stack.append(i)

            elif i == '-':

                op1 = stack.pop()

                stack.append("%s" %x)

                print("{0:^4s} | {1:^4s} | {2:^4s}".format(i,op1,"(-)"))

                x = x+1

                if stack != []:

                    op2 = stack.pop()

                    op1 = stack.pop()

                    print("{0:^4s} | {1:^4s} | {2:^4s}".format("+",op1,op2))

                    stack.append("%s" %x)

                    x = x+1

            elif i == '=':

                    op2 = stack.pop()

                    op1 = stack.pop()

                    print("{0:^4s} | {1:^4s} | {2:^4s}".format(i,op1,op2))

            else:

                op1 = stack.pop()

                if stack != []:

                        op2 = stack.pop()

                        print("{0:^4s} | {1:^4s} | {2:^4s}".format(i,op2,op1))

                        stack.append("%s" %x)

                        x = x+1

exp = input("Enter a valid infix expression  \n")

print("Postfix Notation:")

print(infix\_to\_postfix(exp))

print("\n\nThree Address code:")

T\_A\_C(exp)

print("\n\nQuadruple generation:")

Quadruple(exp)

print("\n\nTriple generation:")

Triple(exp)

Output:

