**NAME: Adwait S Purao**

**UID: 2021300101**

**BRANCH: Comps B**

**BATCH: B2**

**Experiment no. : 5**

**Booth’s algorithm implementation in Python**

def booths\_algorithm():

    multiplicand\_dec = getInput("Mutiplicand")

    multiplier\_dec = getInput("Multiplier")

    multiplicand\_bin = convertDec(multiplicand\_dec)

    multiplier\_bin = convertDec(multiplier\_dec)

    boothsTriumph(multiplicand\_bin,multiplier\_bin)

    print("Decimal Result: " + str(int(multiplier\_dec)\*int(multiplicand\_dec)))

def boothsTriumph(mcand, plier):

    print("multipcand: " + mcand + " multiplier: " + plier)

    product = "00000000" + plier + "0"

    print("Product: " + product)

    print(buildLine(0,mcand,product))

    for i in range(1,9):

        operation = product[len(product)-2:]

        product = perform\_operation(product,mcand,operation)

        print(buildLine(i,mcand,product))

    product = shift(product)

    product = product[9:17]

    print("Product: " + product)

    return

def perform\_operation(product,mcand,operation):

    if operation == "00":

        product = shift(product)

        print("No Operation")

        return product

    elif operation == "01":

        ##Product = Product + mcand

        temp = binAdd(product[0:8],mcand)

        product = temp + product[8:]

        product = shift(product)

        print("Addition")

        return product

    elif operation == "10":

        product = subtraction(product,mcand)

        product = shift(product)

        print("Subtraction")

        return product

    elif operation == "11":

        product = shift(product)

        print("No Operation")

        return product

    else:

        print("An error has occured when choosing operation: Exiting program")

        return 0

def subtraction(product,mcand):

    carry = 0

    prime\_product = product[:8]

    final\_product = ""

    for i in range(len(prime\_product)-1,-1,-1):

        if (mcand[i] == "0" and prime\_product[i] == "0"):

            if (carry == 1):

                final\_product = "1" + final\_product

            else:

                final\_product = "0" + final\_product

        elif (mcand[i] == "1" and prime\_product[i] == "0"):

            if (carry == 1):

                final\_product = "0" + final\_product

            else:

                final\_product = "1" + final\_product

                carry = 1

        elif (mcand[i] == "0" and prime\_product[i] == "1"):

            if (carry == 1):

                final\_product = "0" + final\_product

                carry = 0

            else:

                final\_product = "1" + final\_product

        elif (mcand[i] == "1" and prime\_product[i] == "1"):

            if (carry == 1):

                final\_product = "1" + final\_product

                carry = 1

            else:

                final\_product = "0" + final\_product

        else:

            print("An error has occurred when subtracting: Exiting program")

            return 0

    return final\_product + product[8:]

def shift(product):

    product = "0"+product[:len(product)-1]

    return product

def binAdd(num, num2):

    product = ""

    carry = "0"

    for i in range(len(num)-1,-1,-1):

        if carry == "0":

            if num[i] == "0" and num2[i] == "0":

                product = "0" + product

            elif num[i] == "1" and num2[i] == "1":

                product = "0" + product

                carry = "1"

            else:

                product = "1" + product

        elif carry == "1":

            if num[i] == "0" and num2[i] == "0":

                product = "1" + product

                carry = "0"

            elif num[i] == "1" and num2[i] == "1":

                product = "1" + product

                carry = "1"

            else:

                product = "0" + product

                carry = "1"

    return product

def buildLine(iteration, mcand, product):

    line = "Step: " + str(iteration) + " | Multiplicand: " + mcand + " | Product: " \

    + product[0:8] + " | " + product[8:16] + " | " + product[16]

    return line

def convertDec(dec):

    if int(dec)<0:

        bin = twos\_complement(int(dec))

    else:

        bin = "{0:b}".format(int(dec))

        for i in range(8-len(bin)):

            bin = "0" + bin

    return bin

def getInput(varName):

    #Request input

    boothIn = input('Please enter your ' + varName + ": ")

    while int(boothIn)>127 or int(boothIn)<-128:

        print("Absolute value too big, please try again")

        boothIn = input('Please enter your ' + varName + ": ")

    return boothIn

def twos\_complement(dec):

    adjusted = abs(int(dec) + 1)

    binint = "{0:b}".format(adjusted)

    #Flip bits

    flipped = flip(binint)

    for i in range(8-len(flipped)):

        flipped = "1" + flipped

    return flipped

def flip(string):

    flipped\_string = ""

    for bit in string:

        if bit == "1":

            flipped\_string += "0"

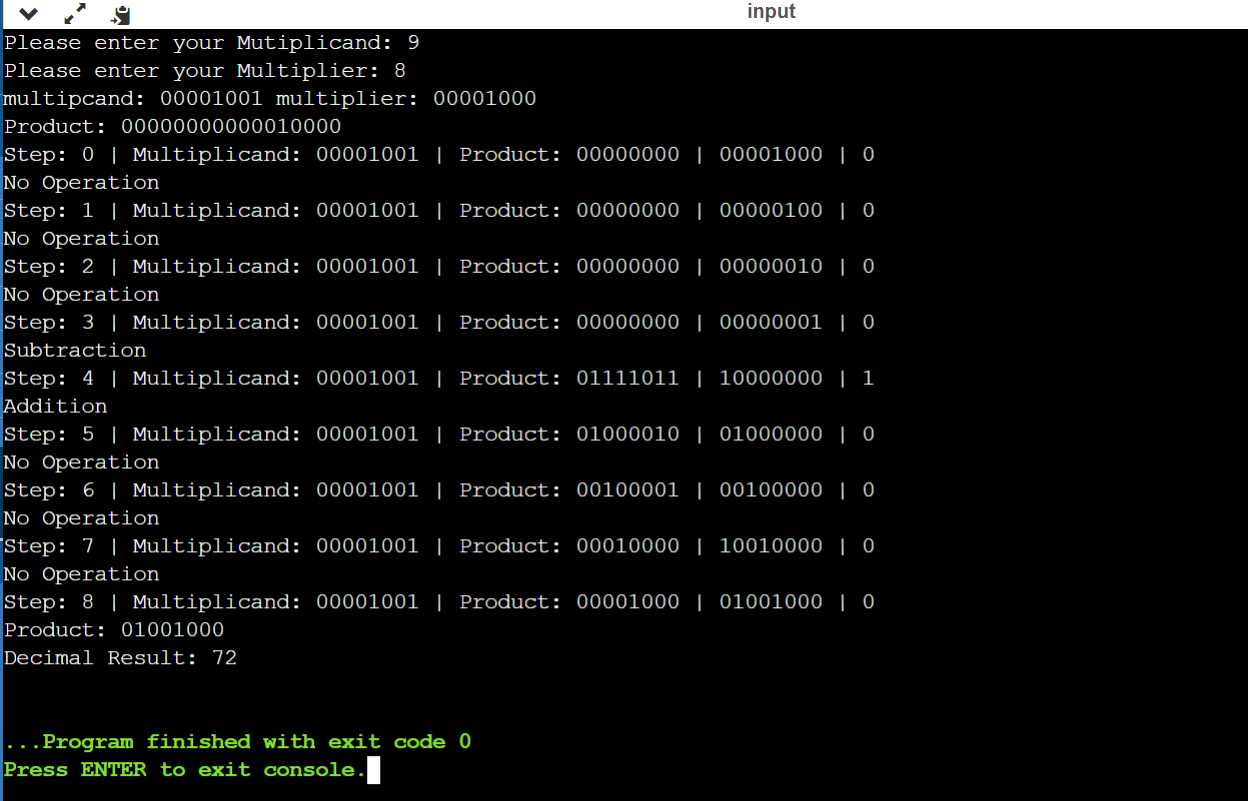
        else:

            flipped\_string += "1"

    return flipped\_string

booths\_algorithm()

**Output:**



**Conclusion:**

In this experiment we learnt about various steps involved in booth’s algorithm and the actual implementation of it with the help of flowchart . We implemented the algorithm in python.