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:

```
input
Please enter your Mutiplicand: 9
Please enter your Multiplier: 8
multipcand: 00001001 multiplier: 00001000
Product: 00000000000010000
Step: 0 | Multiplicand: 00001001 | Product: 00000000 | 00001000 | 0
No Operation
Step: 1 | Multiplicand: 00001001 | Product: 00000000 | 00000100 | 0
No Operation
Step: 2 | Multiplicand: 00001001 | Product: 00000000 | 00000010 | 0
No Operation
Step: 3 | Multiplicand: 00001001 | Product: 00000000 | 00000001 | 0
Subtraction
Step: 4 | Multiplicand: 00001001 | Product: 01111011 | 10000000 | 1
Addition
Step: 5 | Multiplicand: 00001001 | Product: 01000010 | 01000000 | 0
No Operation
Step: 6 | Multiplicand: 00001001 | Product: 00100001 | 00100000 | 0
No Operation
Step: 7 | Multiplicand: 00001001 | Product: 00010000 | 10010000 | 0
No Operation
Step: 8 | Multiplicand: 00001001 | Product: 00001000 | 01001000 | 0
Product: 01001000
Decimal Result: 72


...Program finished with exit code 0
Press ENTER to exit console.
```

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.