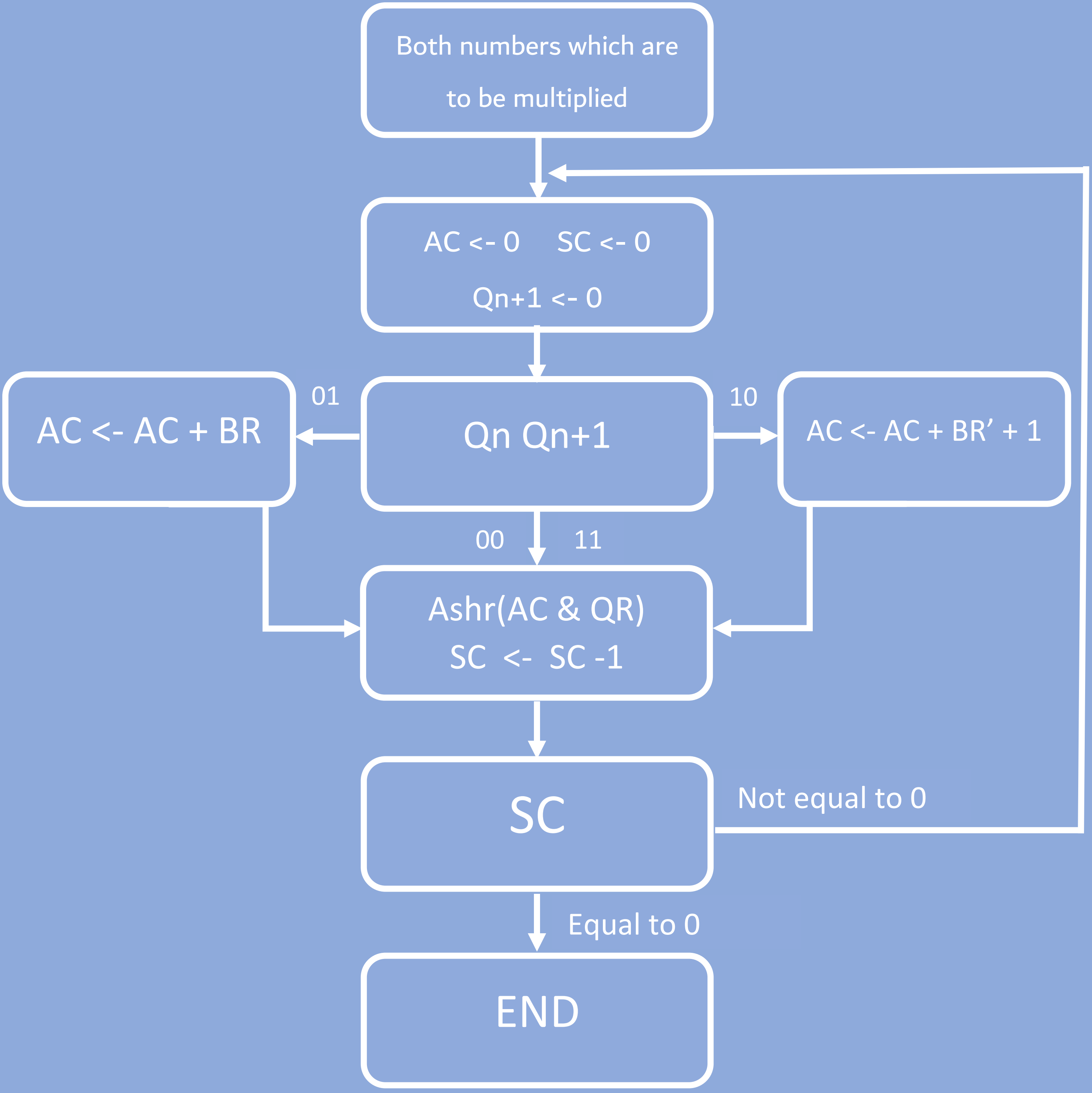
Booth's Algorithm – Deepak Kumar (2019418)

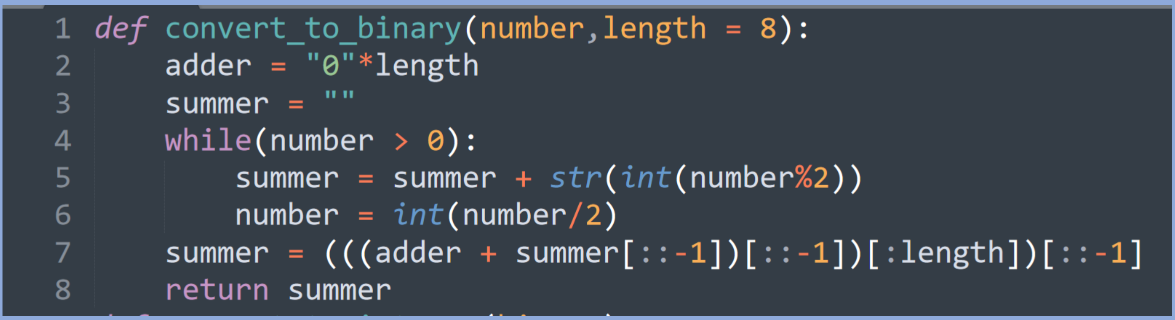
Booth's multiplication algorithm is an algorithm that multiplies two signed binary numbers using its two's complement notation.

Sir Donald Booth designed the algorithm during his research on crystallography in 1950 London. The algorithm is shown below using a chart



⇨ Utility functions explanation in code:

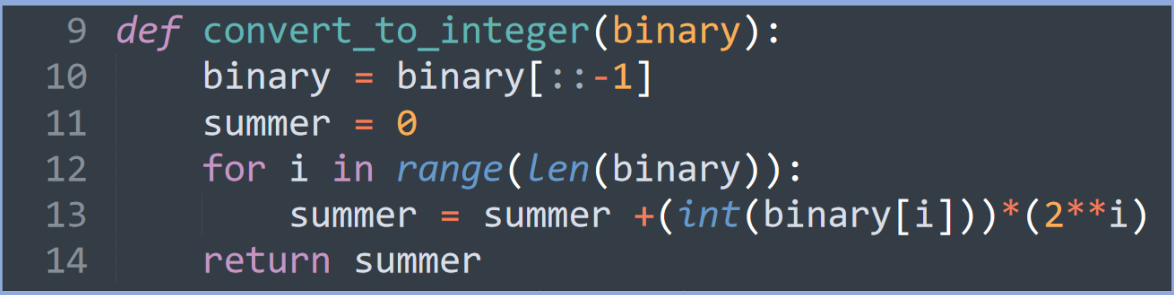
convert_to_binary function :



```python
def convert_to_binary(number,length = 8):
    adder = "0"*length
    summer = ""
    while(number > 0):
        summer = summer + str(int(number%2))
        number = int(number/2)
    summer = (((adder + summer[::-1])[::-1])[:length])[::-1]
    return summer
```

This function uses basic decimal to binary conversion, which is the division with 2 till the end is 0 and at each step adding the remainder and then performing required string formatting and returning the value.

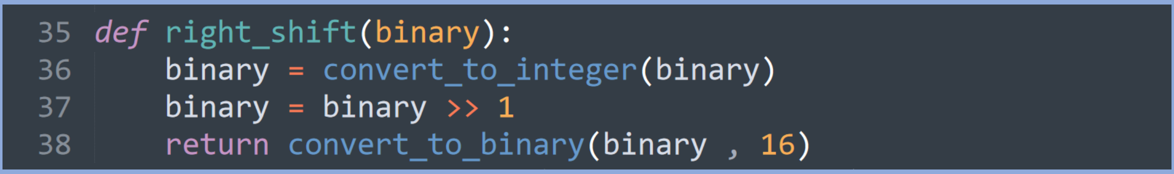
convert_to_integer function :



```python
def convert_to_integer(binary):
    binary = binary[::-1]
    summer = 0
    for i in range(len(binary)):
        summer = summer +(int(binary[i]))*(2**i)
    return summer
```

In this function, first, we reverse the binary string and multiply each positioned 0 or 1 to the power of 2 with their respective position and summing all of it together, giving our decimal number.

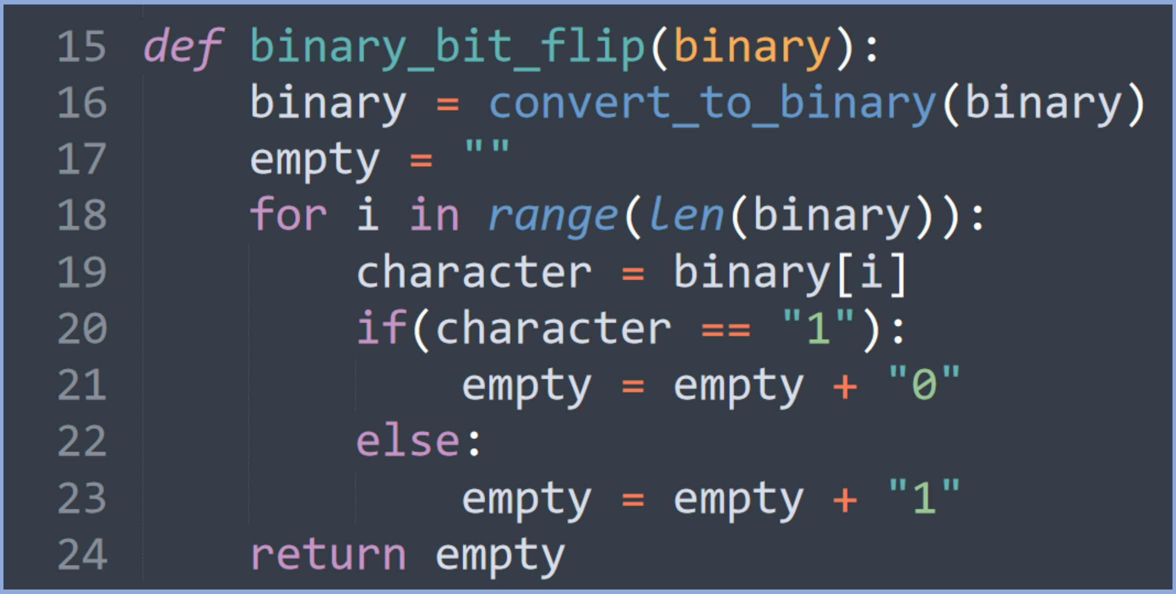
right_shift function :



```python
def right_shift(binary):
    binary = convert_to_integer(binary)
    binary = binary >> 1
    return convert_to_binary(binary , 16)
```

We convert binary to integer and use the bitwise right shift operator on it. After, we convert it to a binary number of 16 bits.

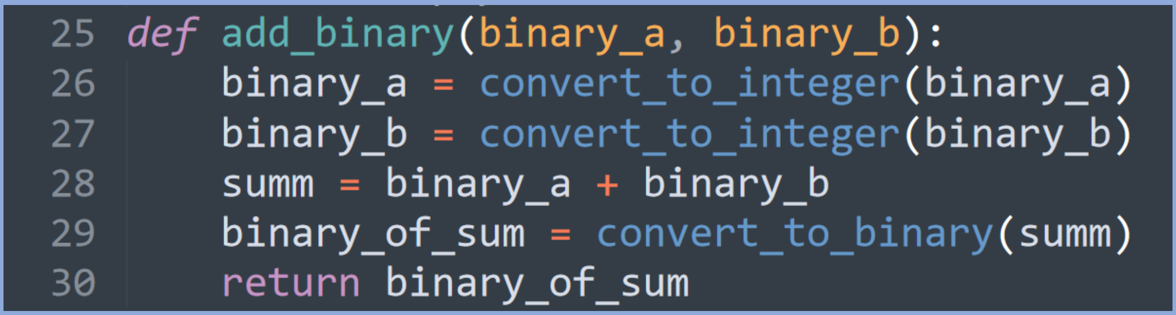
binary_bit_flip function :



```python
def binary_bit_flip(binary):
    binary = convert_to_binary(binary)
    empty = ""
    for i in range(len(binary)):
        character = binary[i]
        if(character == "1"):
            empty = empty + "0"
        else:
            empty = empty + "1"
    return empty
```

In this function, each 0 se converted to 1 and 1 is converted to 0.

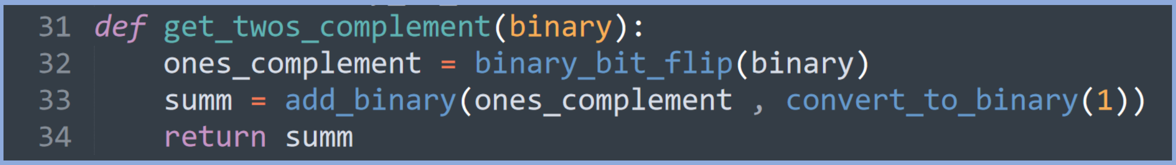
add_binary function :



```python
def add_binary(binary_a, binary_b):
    binary_a = convert_to_integer(binary_a)
    binary_b = convert_to_integer(binary_b)
    summ = binary_a + binary_b
    binary_of_sum = convert_to_binary(summ)
    return binary_of_sum
```

First we convert both binary numbers to integer, and then add both, again convert it to binary and return.

get_twos_complement function :

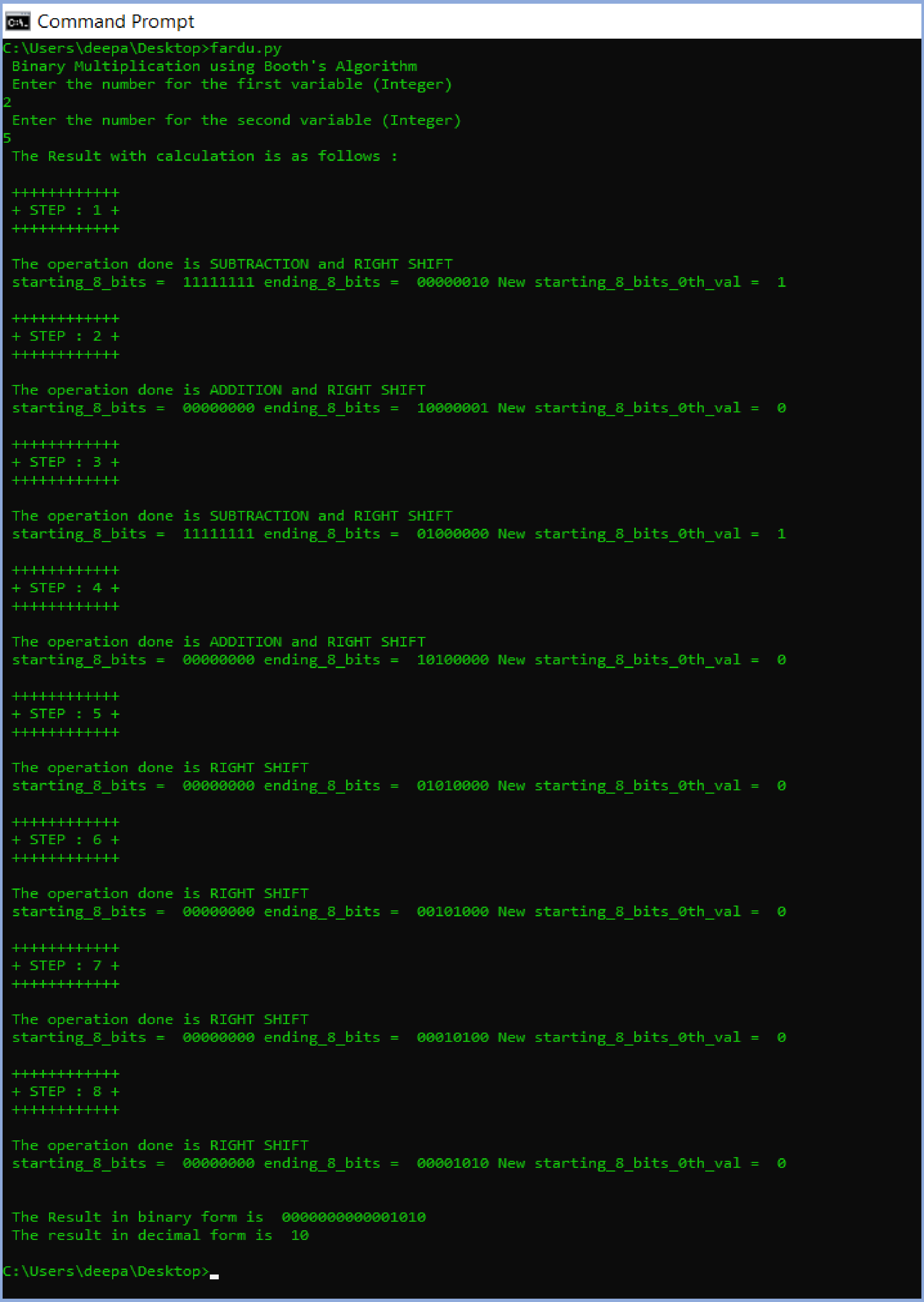


```python
def get_twos_complement(binary):
    ones_complement = binary_bit_flip(binary)
    summ = add_binary(ones_complement , convert_to_binary(1))
    return summ
```

First, we take ones complement of the binary number by flipping each bit, add 1 to it and return their sum.

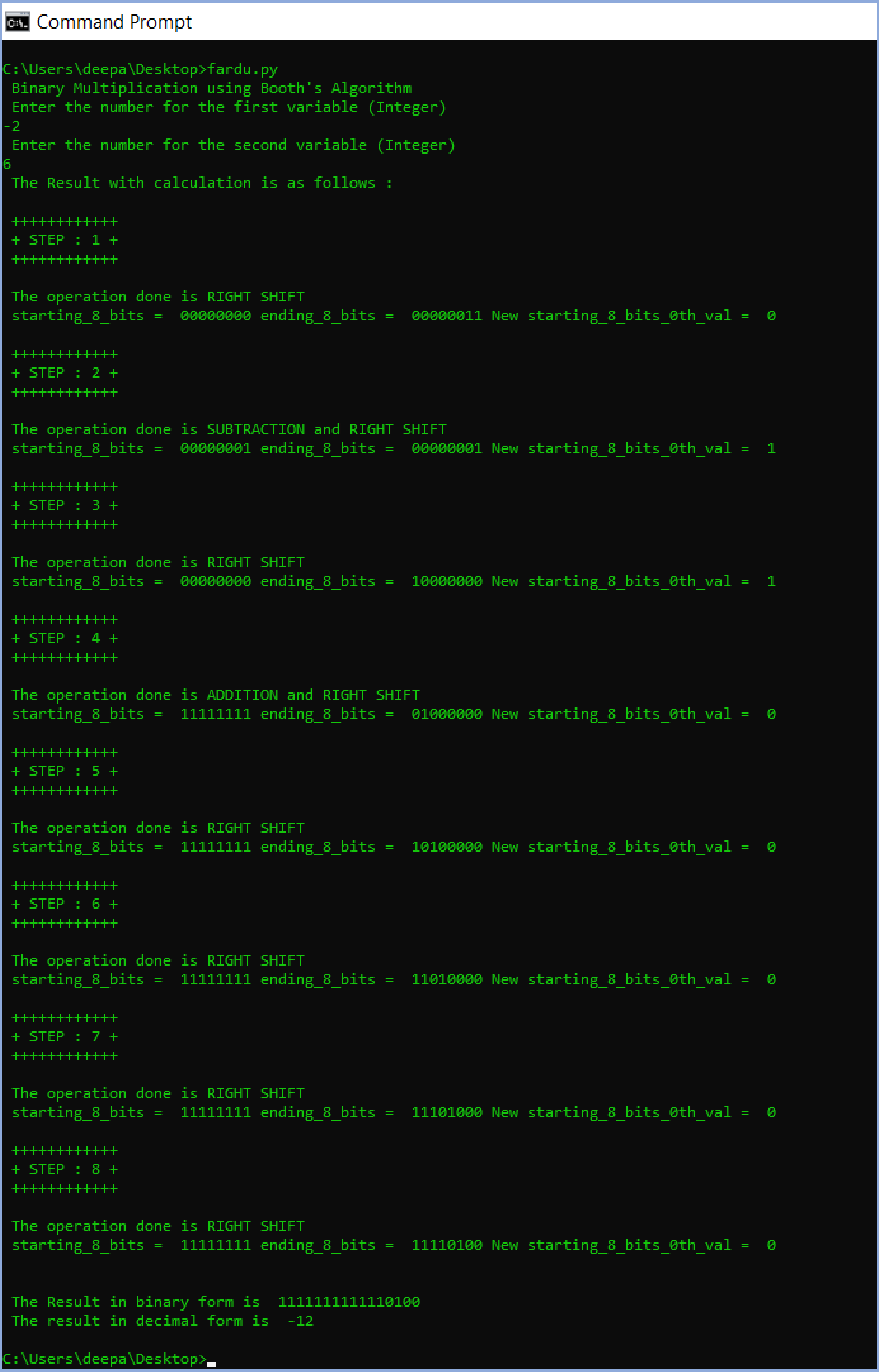
⇨ OUTPUT

O:- FOR TWO POSITIVE NUMBERS



```
C:\Users\deepa\Desktop>fardu.py
 Binary Multiplication using Booth's Algorithm
 Enter the number for the first variable (Integer)
2
 Enter the number for the second variable (Integer)
5
 The Result with calculation is as follows :

 ++++++++++++
 + STEP : 1 +
 ++++++++++++

 The operation done is SUBTRACTION and RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  00000010 New starting_8_bits_0th_val =  1

 ++++++++++++
 + STEP : 2 +
 ++++++++++++

 The operation done is ADDITION and RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  10000001 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 3 +
 ++++++++++++

 The operation done is SUBTRACTION and RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  01000000 New starting_8_bits_0th_val =  1

 ++++++++++++
 + STEP : 4 +
 ++++++++++++

 The operation done is ADDITION and RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  10100000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 5 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  01010000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 6 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00101000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 7 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00010100 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 8 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00001010 New starting_8_bits_0th_val =  0


 The Result in binary form is  0000000000001010
 The result in decimal form is  10

C:\Users\deepa\Desktop>
```

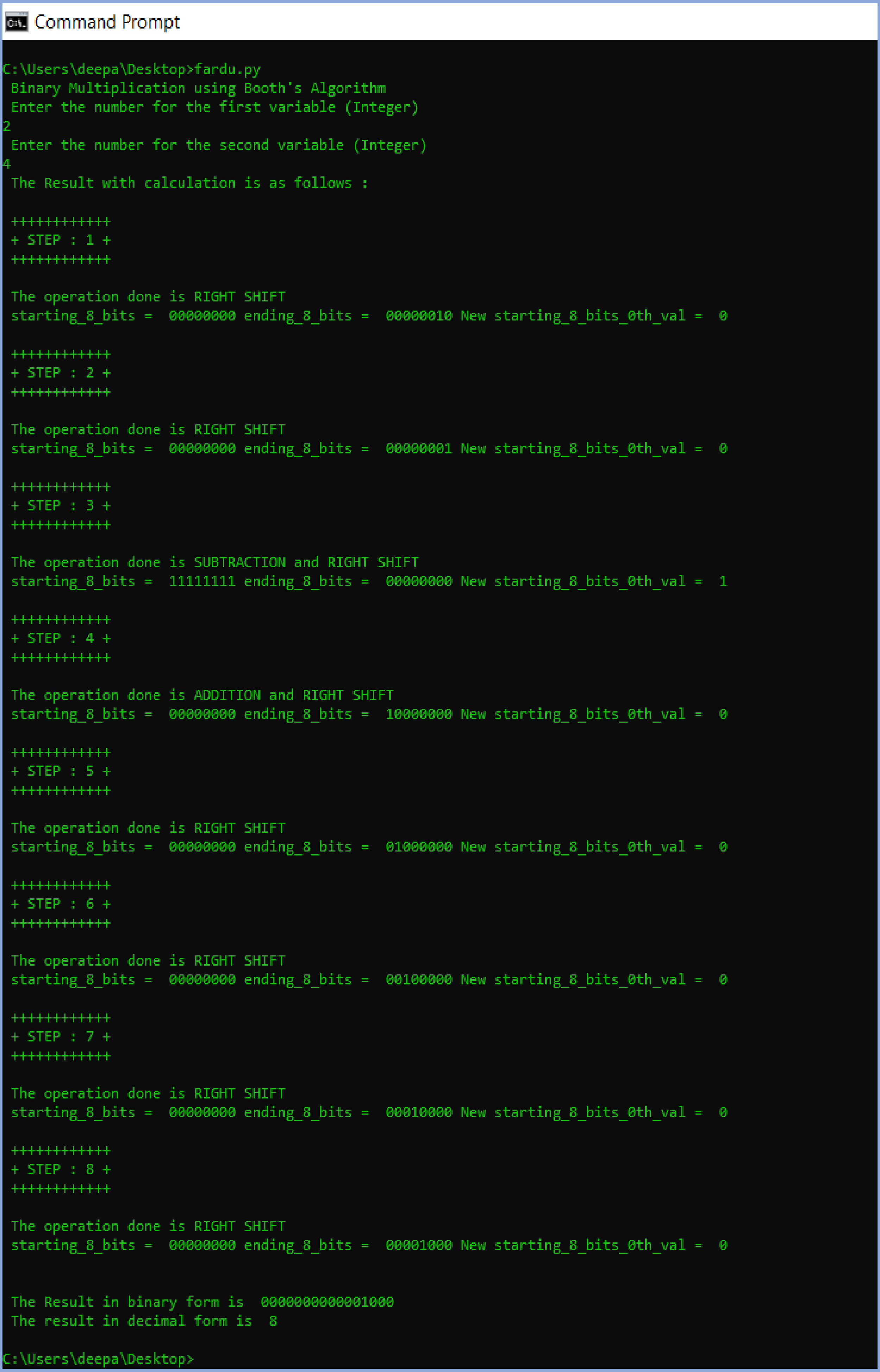
O:- FOR ONE -VE AND ONE +VE NUMBER



Command Prompt

```
C:\Users\deepa\Desktop>fardu.py
 Binary Multiplication using Booth's Algorithm
 Enter the number for the first variable (Integer)
-2
 Enter the number for the second variable (Integer)
6
 The Result with calculation is as follows :

 ++++++++++++
 + STEP : 1 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00000011 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 2 +
 ++++++++++++

 The operation done is SUBTRACTION and RIGHT SHIFT
 starting_8_bits =  00000001 ending_8_bits =  00000001 New starting_8_bits_0th_val =  1

 ++++++++++++
 + STEP : 3 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  10000000 New starting_8_bits_0th_val =  1

 ++++++++++++
 + STEP : 4 +
 ++++++++++++

 The operation done is ADDITION and RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  01000000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 5 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  10100000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 6 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  11010000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 7 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  11101000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 8 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  11110100 New starting_8_bits_0th_val =  0


 The Result in binary form is  1111111111110100
 The result in decimal form is  -12

C:\Users\deepa\Desktop>_
```

O:- FOR ONE -VE AND ONE -VE NUMBER



```
C:\Users\deepa\Desktop>fardu.py
 Binary Multiplication using Booth's Algorithm
 Enter the number for the first variable (Integer)
2
 Enter the number for the second variable (Integer)
4
 The Result with calculation is as follows :

 ++++++++++++
 + STEP : 1 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00000010 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 2 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00000001 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 3 +
 ++++++++++++

 The operation done is SUBTRACTION and RIGHT SHIFT
 starting_8_bits =  11111111 ending_8_bits =  00000000 New starting_8_bits_0th_val =  1

 ++++++++++++
 + STEP : 4 +
 ++++++++++++

 The operation done is ADDITION and RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  10000000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 5 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  01000000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 6 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00100000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 7 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00010000 New starting_8_bits_0th_val =  0

 ++++++++++++
 + STEP : 8 +
 ++++++++++++

 The operation done is RIGHT SHIFT
 starting_8_bits =  00000000 ending_8_bits =  00001000 New starting_8_bits_0th_val =  0


 The Result in binary form is  0000000000001000
 The result in decimal form is  8

C:\Users\deepa\Desktop>
```