



**Module Code & Module Title**

**CS4051NI Fundamentals of Computing**

**Assessment Weightage & Type**  
**60% Individual Coursework**

**Year and Semester**  
**2019-20 Autumn**

**Student Name: Anshul Agarwal**  
**Group: N7**

**London Met ID: 19031269**

**College ID: NP01NT4A190006**

**Assignment Due Date: 8th May 2020**

**Assignment Submission Date:**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.*

## Table of Contents

1. INTRODUCTION .....	5
2. Algorithm .....	6
2.1 Bit Adder .....	6
2.2. Parallel/Cascading Adder Circuit.....	11
2.3 Flow Chart .....	12
2.4 Pseudo code .....	14
3. Data Structure .....	19
4. Test.....	20
Test 1: If we enter the number out of range. ....	20
Test 2: when the string or character value are entered. ....	21
Test 3: when the minimum value and maximum value is entered.....	22
Test 4: When we entered the different kind of signs and symbols. ....	23
Test 5: When we entered the Boolean.....	24
5. Conclusion .....	25
6. Appendix .....	26
Bibliography .....	31

Figure 1: Adder Circuit.....	6
Figure 2: AND Gate .....	7
Figure 3: OR Gate.....	8
Figure 4: XOR Gate .....	9
Figure 5: Sum of bit adder .....	10
Figure 6: Parallel/Cascading Adder Circuit.....	11
Figure 7: Flow Chart .....	12
Figure 8: when the number is not in range .....	20
Figure 9:if we entered the string value .....	21
Figure 10: when we entered the maximum and minimum value .....	22
Figure 11: When we entered the different kind of sings and symbols .....	23
Figure 12: When we entered the Boolean .....	24

Table 1:Truth table AND Gate.....	7
Table 2:Truth Table OR Gate.....	8
Table 3: Truth table for XOR GATE.....	9

## 1. INTRODUCTION

On fifth April we have been assigned to do a sure coursework on which we have to advance a software alongside with the model of byte adder assembled the usage of electronic gates like and gate , or gate, no longer gate and xor gate based totally on model of bit adder. To write a application first we have to specify and algorithm of integer addition based totally on bitwise operation. A appropriate statistics shape is to be processed when running program. As this is our first coursework on python doing it was once quite a bit tough work for us in this modern-day situation where there is no direct contact with our module teachers and tutors teacher, however as the material and tenet supply by them I was once able to complete the coursework a little late after the deadline due to some major problems as described above. The program was once made efficaciously editing and adding some adjustments to suggestions and I hope it is as comparable to the guidelines. However data shape and logic implementation are a little one-of-a-kind from guidelines.

## 2. Algorithm

### 2.1 Bit Adder

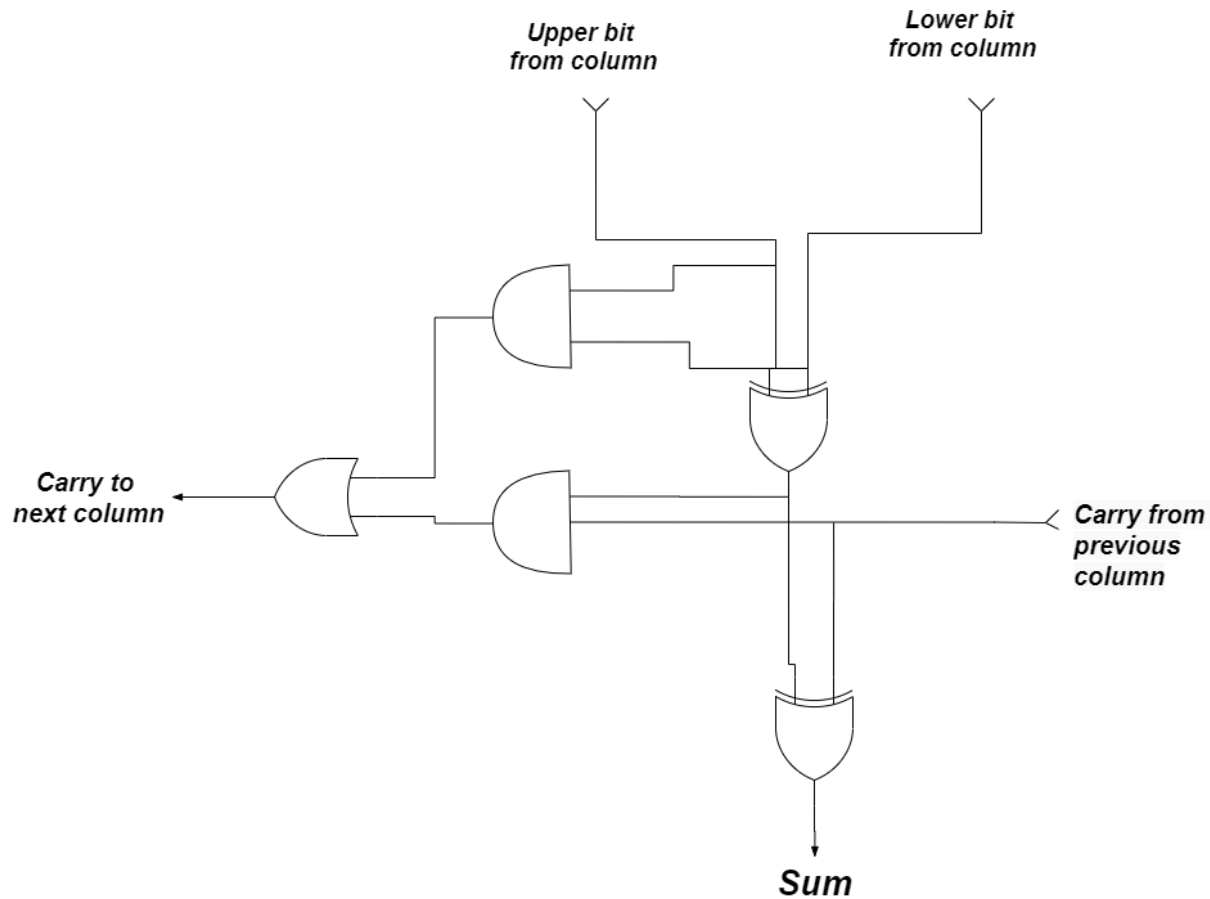


Figure 1: Adder Circuit

The above diagram is used to take out the sum of the two numbers. With the help of arithmetic circuit like arithmetic operation and procedures which can be done with the help of different basic logic gates like AND gate, OR gate, XOR gate. The brief introduction about the entire gate is explained below with the help of truth table:

#### 1. And Gate:

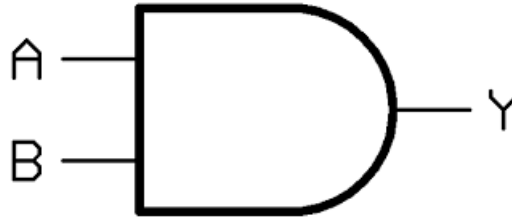


Figure 2: AND Gate

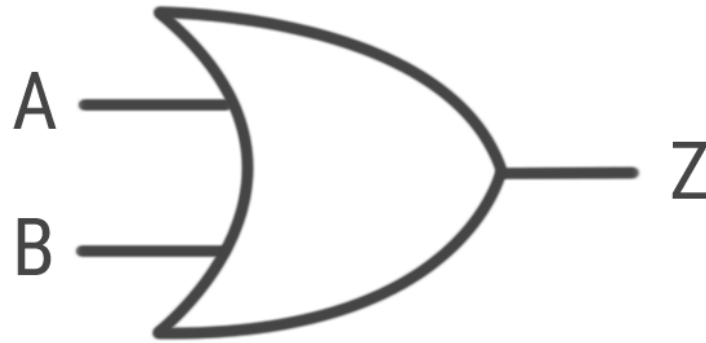
We can give two or more inputs. If all the inputs must be true to get the true values. For e.g. A AND B must be true. If all the inputs are of the other combination the output will result in the false output. The Boolean expression for AND gate with 2 inputs is (A AND B): **A.B**. The truth table for AND gate is shown below:

A	B	output
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth table AND Gate

From the above table we can see that if we give different values the output will also come different. If we keep same value the output will come true.

## 2. OR GATE:

**Figure 3: OR Gate**

We can give two or more inputs. In or gate we have to give at least one value true other value can be false if one of the is true we will gate true output. If the both values are true also the output will be true only. The Boolean expression for or gate with 2 inputs (A or B):  $X+Y$ . The truth table for or gate is shown below:

A	B	output
0	0	0
0	1	1
1	0	1
1	1	1

**Table 2:Truth Table OR Gate**

From the above table we can conclude that if the both input are same the value is true. if one of the output is true the output will be true. If the both value are false the output will be false.



## 3. XOR GATE:

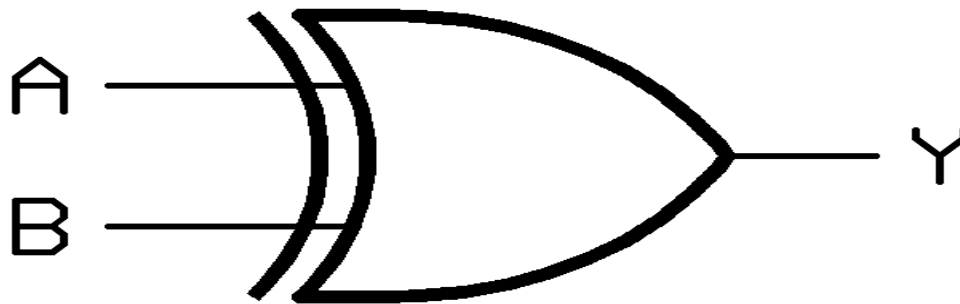


Figure 4: XOR Gate

We can give two or more inputs. In xor gate we have only one input true the output will be true and if we give both the input true the output will come false. The Boolean expression for xor gate with 2 inputs (A XOR B):  $\bar{A}.B + A.\bar{B}$ . The truth table for xor gate is shown below:

A	B	output
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Truth table for XOR GATE

From the above table we can conclude that if we keep same values in the input the output will come false output. If we keep true input the output will come true.

In the fig:2 if few keep the values in the full bit adder operation for eg  $A=0, B=1$  and  $C_{in}=1$ . If we keep the in circuit the sum will 0.

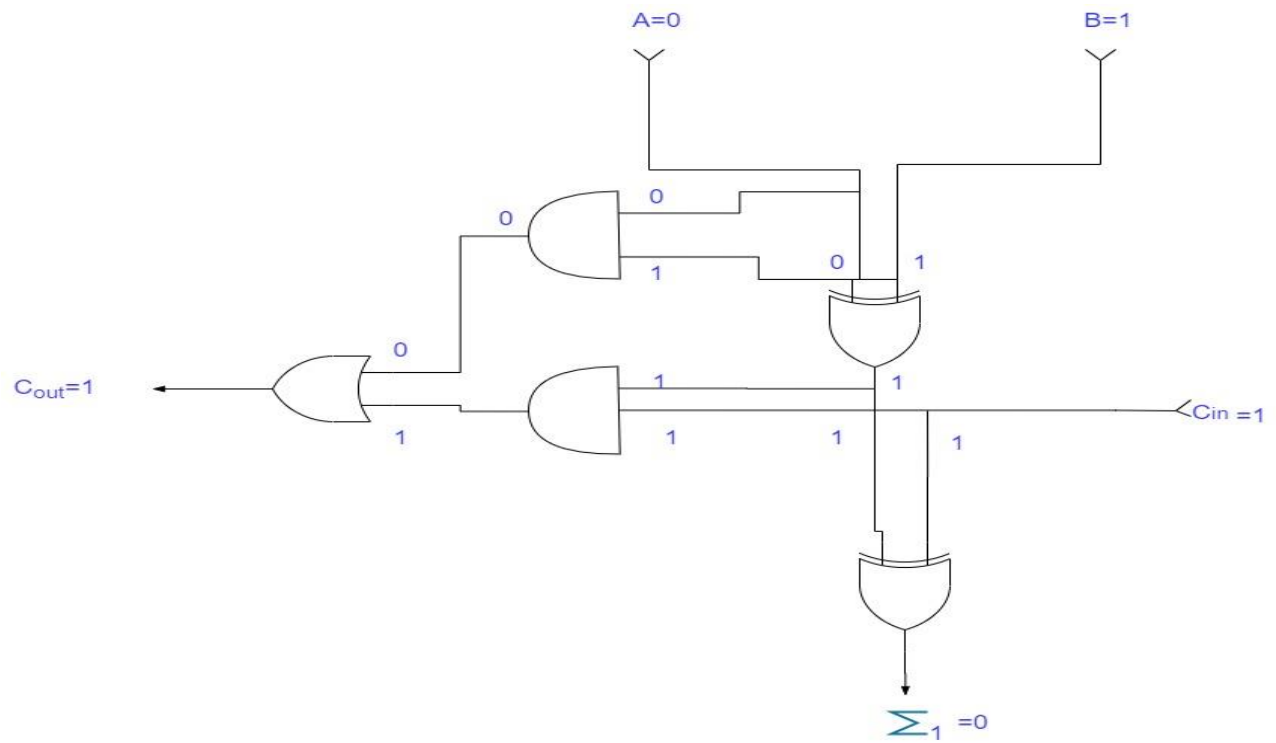


Figure 5: Sum of bit adder

## 2.2. Parallel/Cascading Adder Circuit

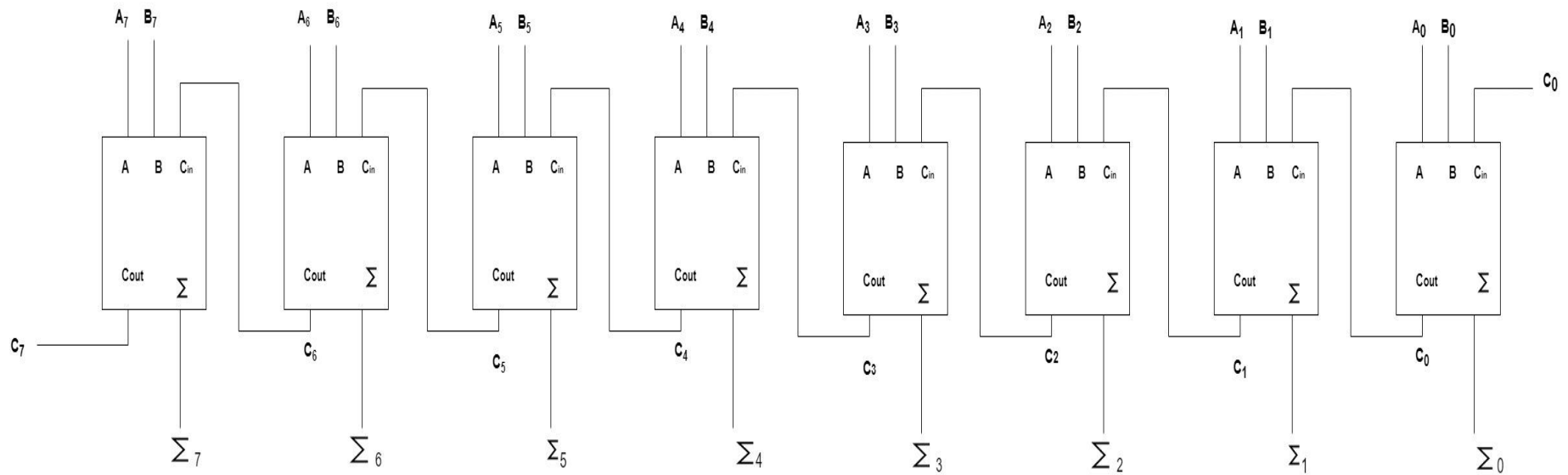


Figure 6: Parallel/Cascading Adder Circuit

This is a full adders are combined into parallel adders which helps us to add binary numbers with multiple bits. A 8-bit adder is shown above which can add 4-bit and 8 bit also for example adding 60(111100) to 63(111111) or 250(11111010) to 251 (11111011)

## 2.3 Flow Chart

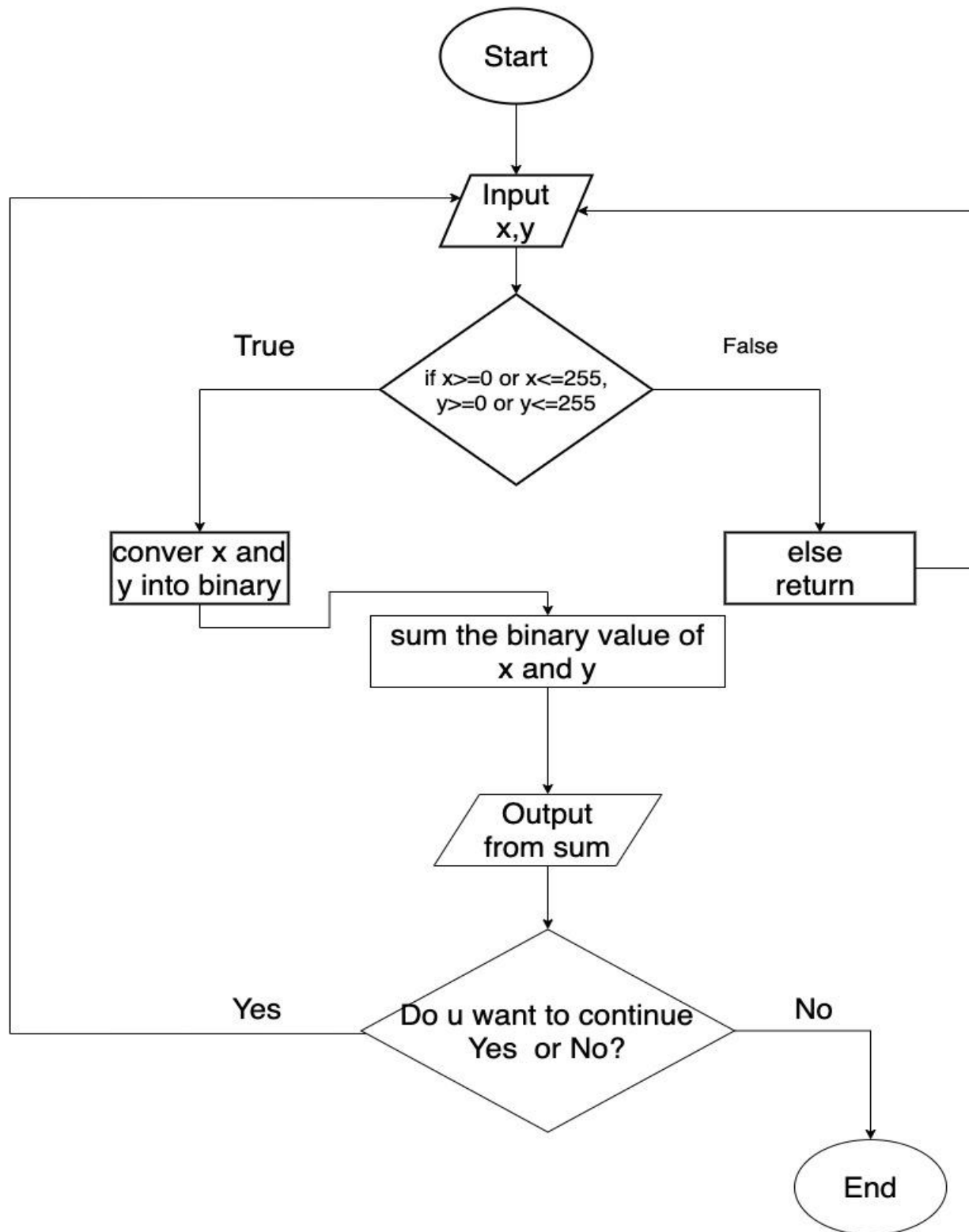


Figure 7: Flow Chart

At first we have to enter the value of x and y. Then the value is compared and checked that they are in range or not and they are only integer value if they are not in range or any kind of the symbol or string word then the it tell again to input the number. After comparing then the value of x and y are to the binary conversion in the form of 0 and 1. after the binary conversion then the values are send for the sum of the both number x and y. then the value of x and y are displayed. After the output the message is asked again that do you want to continue or not .if we enter yes the entire process take place again. If we pressed no then the code comes to an end.

## 2.4 Pseudo code

**From** binaryconversion **import** binaryconversation

**From** binnaryadder **import** binaryadders

**Output** Binnary Adder

Runs = " yes"

**While** runs == "yes" **or** runs==" y"

**try:**

**input** x

**if** (x>255)

**output** the number must be between 0 and 255

**else:**

ins=**True**

**except:**

**output** please enter the valid number

ins=**False**

**while** ins==**False:**

**try:**

**input** y

**if** y >255

**output** the number must be between 0 and 255

**else:**

ins=**True**

**except:**

**output** please input the valid number

**output** "the binary value of {} is:{}\n".format (x,binaryconversations(x))

**output** "the binary value of {} is:{}\n".format (x,binaryconversations(y))

a=binaryconversations(x)

b=binaryconversation(y)

sums1=binaryadders(**str**(a),str(b))

**output** "the sum of {} and is:{} \n".format(x,y,sums1)

**input** runs

**Function** notgate(a):

**If** a==0:

**Return** 1

**else:**

**Return** 0

**Function** orgate(a,b):

**if** a==b==0:

**return** 0

**else:**

**return 1**

**Function** andgate(a,b):

if a==b==1:

**return 1**

**else:**

**return 0**

**function** xorgate(a,b):

a1=andgate(a,notgate(b))

b1=andgate(b,notgate(a))

**return** int(orgate(a1,b1))

**function** halfadder(a,b):

x=xorgate(a,b)

y=andgate(a,b)

**return**(x,y)

**function** fulladder(a,b,c=0):

sum1,c1=halfadder(a,b)

sum2,c2=halfadder(sum1,c)

**return** (sum2,orgate(c1,c2))

**function** binaryadders(a,b):

**if** a>b:

tmp=a



```
a=b

b=tmp

c=0

results=""

for i in range(len(a)-1,-1,-1):

    sums1,c= fulladder(int(a[i]),int(b[i],c))

    results+=str(sums1)

results=results[::-1]

l=len(results)

if len(results)<8:

    for i in range(l,8):

        results="0"+results

return results

function binaryconversations(v):

    v=int(v)

    bits=[]

    actualbinary=[]

    actualbinarynum=""

    ls=0

    while v!=0:

        reminders=v%2
```

```
bits.append(reminders)

v=v//2

for i in range(len(bits)-1,-1,-1):

    actualbinary.append(bits[i])

    actualbinarynum=actualbinarynum+str(bits[i])

    ls= len (actualbinarynum)

    if len(actualbinarynum)<8:

        for i in range(ls,8):

            actualbinarynum="0"+actualbinarynum

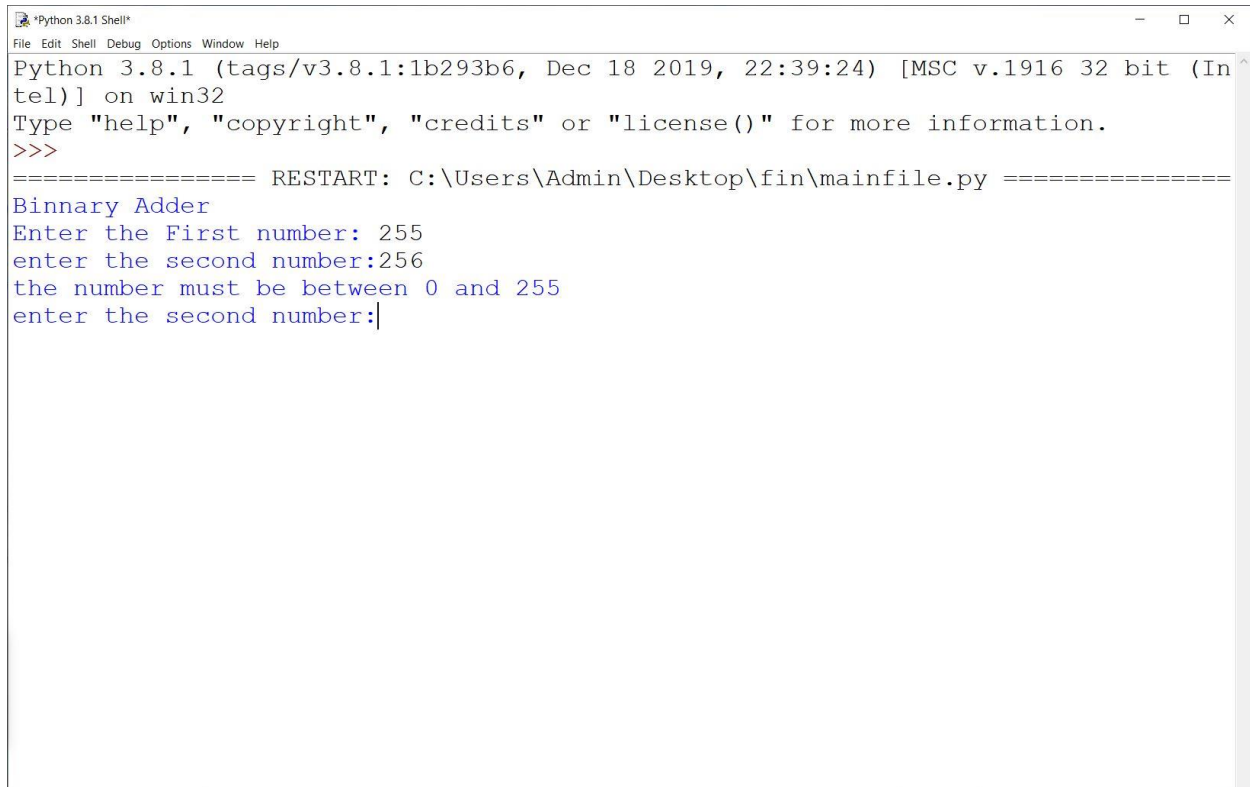
return actualbinarynum
```

### 3. Data Structure

While growing this coursework I have realized pretty fascinating matters about python like function, tuples, lists, strings so I have applied the thinking here. In the main file I have used tuples in the program, string, lists. I have realized these from a website. Tuples are used in programming for grouping of data, lists are used for accumulating the data in order and list can be used interior of listing which is known as nested and is denoted by means of empty or [ ] this symbol. Algorithm is additionally used and a event handling thinking is used and importing of specific characteristic in one single file is completed at ultimate the software is compiled and run which runs virtually how we desired them to run.

## 4.Test

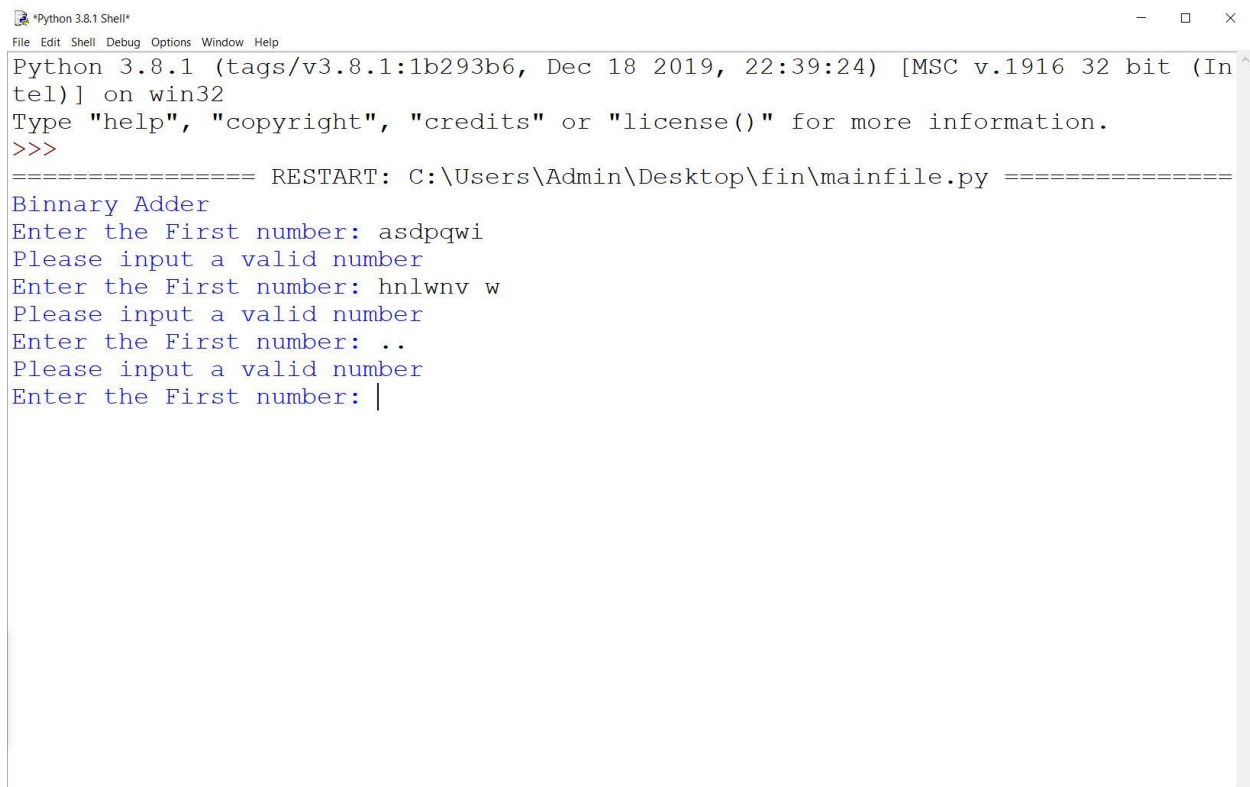
**Test 1:** If we enter the number out of range.



```
Python 3.8.1 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\fin\mainfile.py =====
Binnary Adder
Enter the First number: 255
enter the second number:256
the number must be between 0 and 255
enter the second number:|
```

Figure 8: when the number is not in range

When we input the wrong number or enter the number of the range then it goes on asking please enter the valid number until the we enter the write number

**Test 2: when the string or character value are entered.**

```
Python 3.8.1 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\fin\mainfile.py =====
Binnary Adder
Enter the First number: asdpqwi
Please input a valid number
Enter the First number: hnlwnv w
Please input a valid number
Enter the First number: ..
Please input a valid number
Enter the First number: |
```

Figure 9: if we entered the string value

When we input the wrong number or enter any kind of the character or string value then the it goes on asking please enter the valid number until the we enter the write number.

**Test 3: when the minimum value and maximum value is entered.**

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\fin\mainfile.py =====
Welcome to Binnary Adder
Enter the First number: 0
enter the second number:255
The binary value of 0 is:00000000

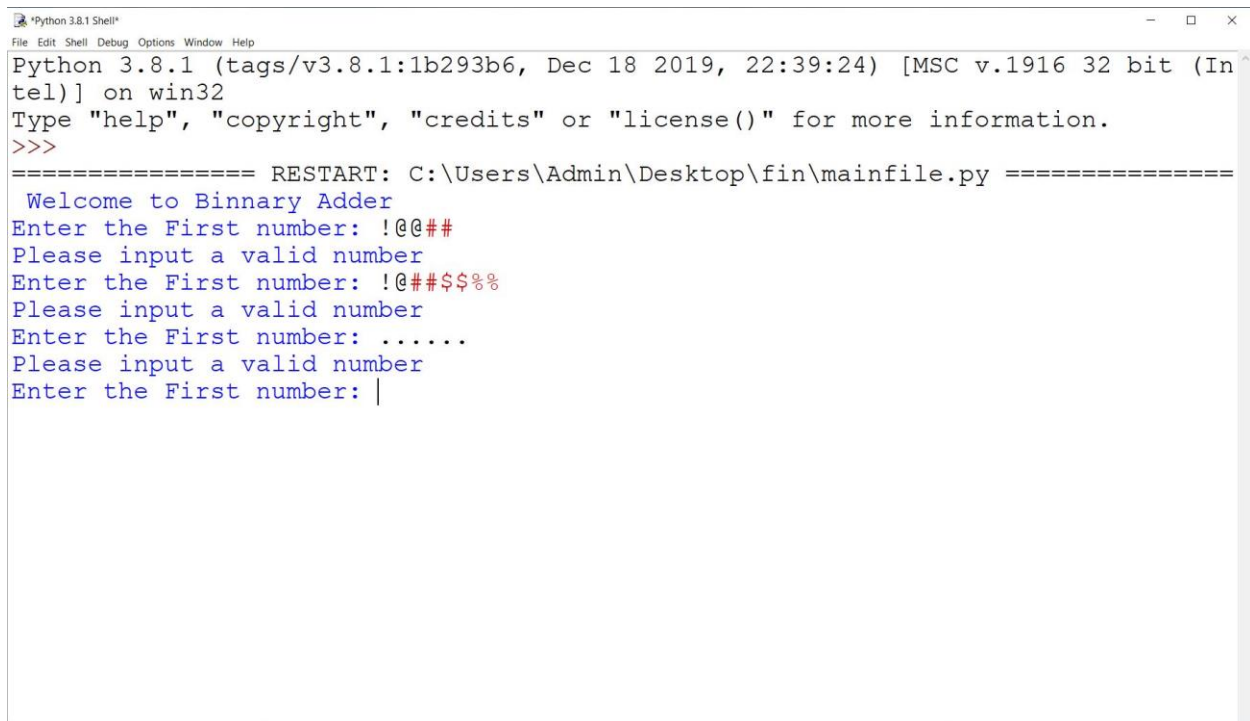
The binary value of 255 is:11111111

the sum of 0 and 255 is:11111111

Do you want to continue? yes/no:|
```

**Figure 10: when we entered the maximum and minimum value**

When we enter the maximum and minimum then two number will be added .

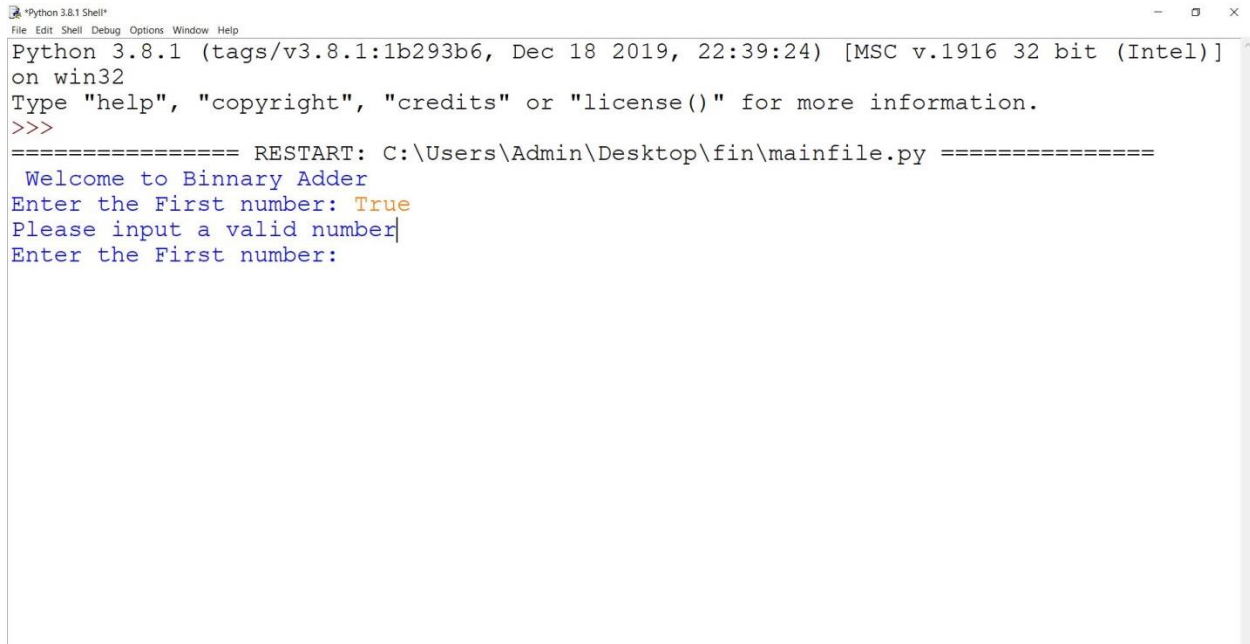
**Test 4: When we entered the different kind of sings and symbols.**

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\fin\mainfile.py =====
Welcome to Binnary Adder
Enter the First number: @@##
Please input a valid number
Enter the First number: @##$$%%
Please input a valid number
Enter the First number: .....
Please input a valid number
Enter the First number: |
```

**Figure 11: When we entered the different kind of sings and symbols**

When we input the wrong number or enter any kind of the symbols or sign then the it goes on asking please enter the valid number until the we enter the write number

## Test 5: When we entered the Boolean



```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\fin\mainfile.py =====
Welcome to Binnary Adder
Enter the First number: True
Please input a valid number|
Enter the First number:
```

Figure 12: When we entered the Boolean

When we input the wrong number or enter any kind of the Boolean then the it goes on asking please enter the valid number until the we enter the write number



## 5. Conclusion

However completing this coursework used to be a lot tougher then it appears due to the fact we may come across any varieties of hassle at any time and in my application I encountered a a lot of bugs too. But with the assist of instructor tips I was once capable to entire this coursework. At the quit of this coursework I realized what really I learned from this coursework was once a lot more and I additionally realized that all know-how that I acquire in my lecture and lab was once carried out here too. To be precise I have gained quite excellent know-how about the way of programming, how to use functions, conditions, string, tuples, lists and algorithm. Furthermore I have realized about the use of gates for situation and the usage of that case for making flowchart. Plus I have discovered about 8 – bit adder and how these bits features and how to use gates in programming style. I sense happy about gaining such expertise from this coursework which used to be now not handy and I encountered a plenty of error but now I would like to say that I would like to acquire comparable extra fascinating coursework in the upcoming future for improving my knowledge.

## 6Appendix

```
def binaryconversations(v):
```

```
    v=int(v)
```

```
    bits=[]
```

```
    actualbinary=[]
```

```
    actualbinarynum=""
```

```
    ls=0
```

```
    while v!=0:
```

```
        reminders=v%2
```

```
        bits.append(reminders)
```

```
        v=v//2
```

```
    for i in range(len(bits)-1,-1,-1):
```

```
        actualbinary.append(bits[i])
```

```
        actualbinarynum=actualbinarynum+str(bits[i])
```

```
        ls= len(actualbinarynum)
```

```
    if len(actualbinarynum)<8:
```

```
        for i in range(ls,8):
```

```
            actualbinarynum="0"+actualbinarynum
```

```
    return actualbinarynum
```

#This module calculate the addition of two binary number

#basic not gates

def notgate(a):

if a==0:

return 1

else:

return 0

#basic or gate

def orgate(a,b):

if a==b==0:

return 0

else:

return 1

#basic and gate

def andgate(a,b):

if a==b==1:

return 1

else:

return 0

#basic xor gate

def xorgate(a,b):

```
a1=andgate(a,notgate(b))

b1=andgate(b,notgate(a))

return int(orgate(a1,b1))

def halfadder(a,b):

    x=xorgate(a,b)

    y=andgate(a,b)

    return(x,y)

def fulladder(a,b,c=0):

    sum1,c1=halfadder(a,b)

    sum2,c2=halfadder(sum1,c)

    return (sum2,orgate(c1,c2))

def binaryadders(a,b):

    if a>b:

        tmp=a

        a=b

        b=tmp

    c=0

    results=""

    for i in range(len(a)-1,-1,-1):

        sums1,c = fulladder(int(a[i]),int(b[i]),c)

        results+=str(sums1)
```

```
results=results[::-1]

l=len(results)

if len(results)<8:

    for i in range(l,8):

        results="0"+results

return results


from binaryconversion import binaryconversations

from binnaryadder import binaryadders

#from binnaryadder import decimltobinary

print(" Welcome to Binnary Adder ")

runs="yes"

while runs=="yes" or runs=="y":

    ins=False

    while ins==False:

        try:

            x=int(input("Enter the First number: "))

            if(x<0 or x>255):

                print("the number must be between 0 and 255")

            else:

                ins=True
```

```
except:

    print("Please input a valid number")

ins=False

while ins==False:

    try:

        y=int(input("enter the second number:"))

        if (y<0 or y>255):

            print("the number must be between 0 and 255 ")

        else:

            ins=True

    except:

        print("please input a valid number")

print("The binary value of {} is:{}\n".format(x,binaryconversations(x)))

print("The binary value of {} is:{}\n".format(y,binaryconversations(y)))

a=binaryconversations(x)

b=binaryconversations(y)

sums1=binaryadders(str(a),str(b))

print("The sum of {} and {} is:{}\n".format(x,y,sums1))

runs=input("Do you want to continue? yes/no:")

(Alder, 2010) (Peter Wentworth, 2012)
```

## **Bibliography**

Alder, G. (2010) *drew.io* [Online]. Available from: <https://app.diagrams.net/> [Accessed 1 May 2020].

Peter Wentworth, J.E.A.B.D.a.C.M. (2012) In *How to Think Like a Computer Scientist*.