



**slington college**  
(इस्लिङ्टन कलेज)

**Module Code & Module Title**

**CS4051NI Fundamentals of Computing**

**Assessment Weightage & Type**

**60% Individual Coursework**

**Year and Semester**

**2019-20 Autumn**

**Student Name: Himanshu Pandey**

**Group:N7**

**London Met ID:19031311**

**College ID:NP01NT4A190131**

**Assignment Due Date:7<sup>TH</sup> JUNE 2020**

**Assignment Submission Date: 8<sup>TH</sup> JUNE 2020**

*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:

### Contents

8. Introduction .....	25
\ Pseudocode	
Algorithm	
Data Structure	
Testing	
Conclusion	
Appendix .....	1

## 1.List of Figures

Figure 1Algorith of byte adder .....	10
Figure 2Flowchart of the program .....	11
Figure 3diagram of 8 bit adder .....	12
Figure 4testing minimum value .....	13
Figure 5testing boolean as input .....	14
Figure 6testing random character.....	14
Figure 7testing greater than maximum value .....	15
Figure 8testing rndom numbers .....	15

## 1. 2. Introduction

On 5th April we were assigned to do a certain coursework on which we have to develop a program along with the model of byte adder assembled using electronic gates like and gate , or gate, not gate and xor gate based on model of bit adder. As this is our first coursework on python doing it was quite a bit hard work for us in this current situation where there is no direct contact with our module teachers and tutors teacher, but as the material and guideline provide by them I was able to complete the coursework a little late after the deadline due to some major problems as described above. To write a program first we have to specify and algorithm of integer addition based on bitwise operation. A suitable data structure is presented to be processed when running program. but as the material and guideline provide by them I was able to complete the coursework was made successfully modifying and adding some changes to guidelines and I hope it is as similar to the guidelines. However data structure and logic implementation are a little from different guidelines.

## Pseudocode

```
("**Welcome to the Binary Adder**")
```

```
input = "yes"
```

**while** input == "True" or input == "T":

**output** = False

**while** output == False:

**try**:

            H=int(input("Please enter the first number: "))

**if** H>255:

                print("The number should be in range of 0 and 255")

**else**:

                output = True

**except**:

            ("Please Enter a valid input!")

**output** = False

while out == False:

**try**:

        N= int(input("Please enter the second number: "))

**if** N>255:

            print("The number must be in range of 255: ")

**else**:

            out= True

**except**:

        ("please input a valid number!")

    ("The binary value of {} is: {}\n".format(H,binary\_conversion(H)))

    ("The binary value of {} is: {}\n".format(N,binary\_conversion(N)))

    Y=binary\_conversion(H)

    Z=binary\_conversion(N)

    sum1= binaryAdder(str(Y),str(Z))

    ("The sum of {} and {} is: {}\n".format(H,N,sum1))

**def** binaryConversion(n):

    bit=[]

```
Binary_Number=""

counter=0
while counter!=8:
    reminder=n%2
    bit.append(reminder)
    n=n//2
    counter=counter+1
return bit
def reverse(bit):
    Binary=[]
    Binary_Number=""
    for i in range(len(bit)-1,-1,-1):
        Binary.append(bit[i])
        BinNumber=Bin_Number+str(bit[i])
    return Bin_Number

#not gate
def not_gate(y):
    if Y==0:
        return '1'
    else:
        return '0'

#xor gate
def xor_gate(Y,Z):
    Y1 = and_gate(Y, not_gate(Z))
    Z1 = and_gate(Z, not_gate(Y))
    return int(or_gate(Y1,Z1))

#and gate
def and_gate(Y,Z):
```

```

if (Y=='1' and Z=='1'):
    return '1'
else:
    return '0'

```

#or gate

```

def or_gate(Y,Z):
    if(Y==Z=='0'):
        return '0'
    else:
        return '1'

```

#byte adder

```

#def binaryAdder(Y,Z):
Def XYZ(binaryNumber1,binaryNumber2):
    #converting binarynumber to string
    str1=str(binaryNumber1)
    str2=str(binaryNumber2)

```

#converting to 8 digit bytes

```

for i in range(0,(8- len(str1) ),+1) :
    str1= '0'+ str1'

```

```

for i in range(0,(8 - len(str2)),+1) :
    str2= 0+ str2

```

```

    _sum1=""

```

```
carry='0'
```

```
for i in range (7,-1,-1):
```

```
    Y,Z=str1[i],str2[i]
```

```
    _sum1= _xor(xor(X,Y),_and(_xor(Y,Z),carry))
```

```
if(carry==0):
```

```
    return _sum
```

```
else:
```

```
    ("sorry,sum is too much huge for stored in byte")
```

```
#Byte Adder('1','1')
```

```
Loop=True
```

```
while Loop==True:
```

```
    stringfirst=False
```

```
    while stringfirst==False:
```

```
        try:
```

```
            binaryNumber1=int(input("Enter the decimal number first:"))
```

```
            binaryNumber1=valid(binaryNumber1)
```

```
            ("\n")
```

```
            stringfirst=True
```

```
        except:
```

```
            ("Invalid Input!")
```

```
            ("\n")
```

```
            ("please enter int value")
```

```
    stringsecond=False
```

```
    while stringsecond==False:
```

```
        try:
```



```
binaryNumber2=int(input("Enter the decimal number second:"))
binaryNumber2=valid(binaryNumber2)
("\n")
stringsecond=True
except:
    print("Invalid Input!")
    ("\n")
    ("please enter int value")
reversefirst=binaryConversion(binaryNumber1)
reversesecond=binaryConversion(binaryNumber2)

first=reverse(reversefirst)
second=reverse(reversesecond)
XYZ(first,second)
result = XYZ(first,second)
```

### 3.Algorithm

This is the algorithm used to develop the programs. Here two bit from column are taken (Higher and lower bit) and And gate , Xor gate and or gate is used for data processing, adding and giving out the appropriate output.

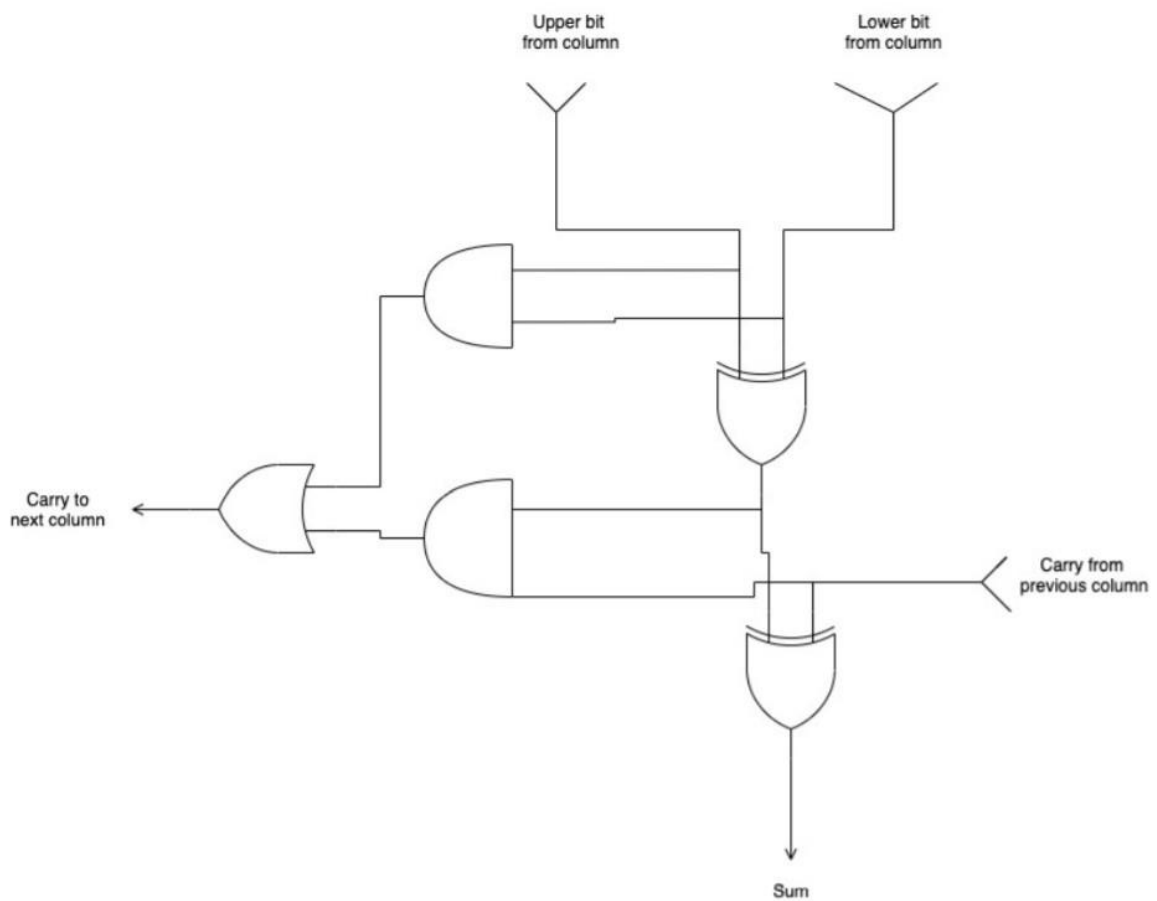


Figure 1Algorithm of byte adder

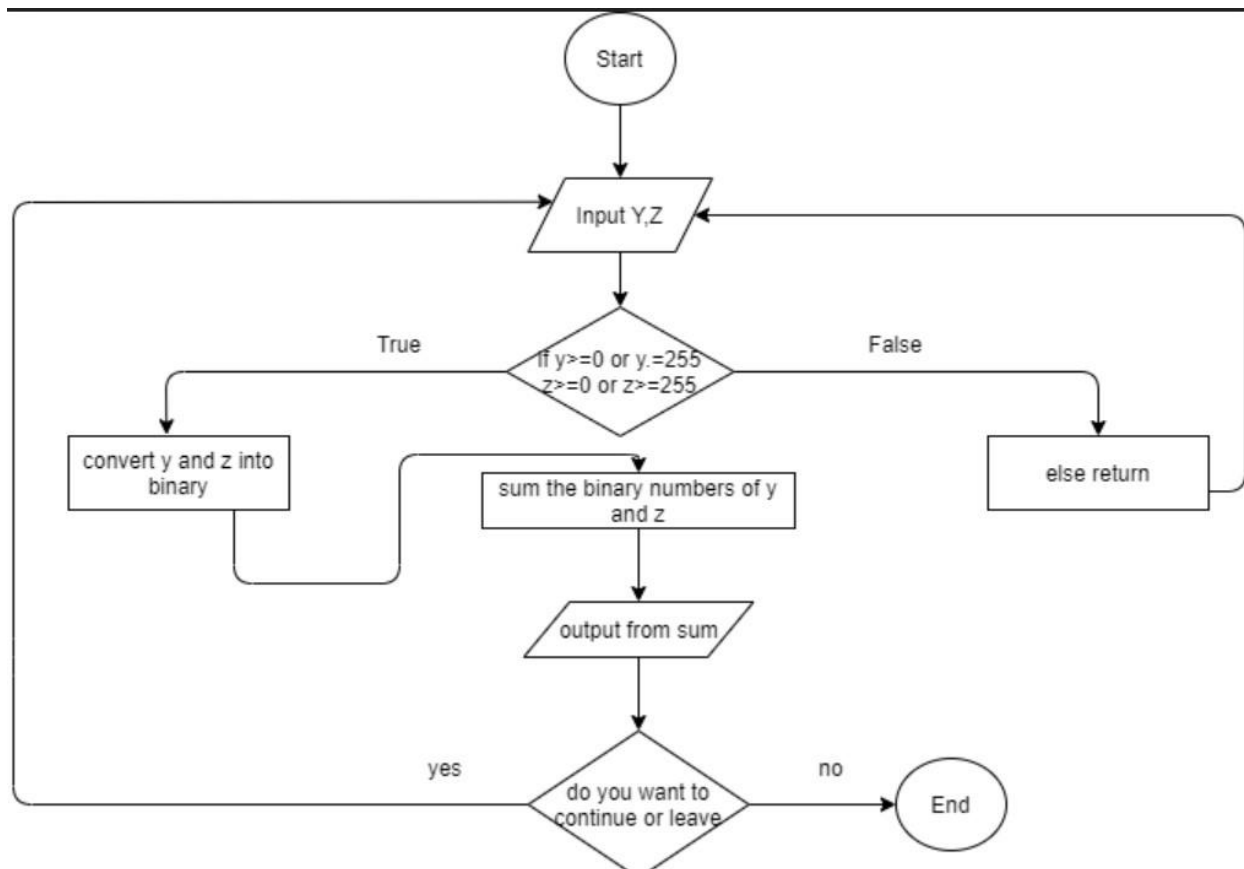


Figure 2 Flowchart of the program

The picture above is the flowchart of the program that I have made. This flowchart describes in summary about how the program is made and how it works so in this flowchart first of all an input is taken from the user and validating of the input value is performed after the validation is done there is two cases , if the input value is outbound the limit i.e. false for the given condition then the program returns to the start where it takes the input again but if the condition is True then the program process the given input values and convert it into binary form further before output the processed binary number is added in binary form and the result is printed. After the result there will be a decision case where the user will be asked whether they want to continue to do more conversion and addition or not if in the prompt the user inputs yes or y then the program executes

and runs in a loop from the beginning taking input but if the input is no the program leave and exits.

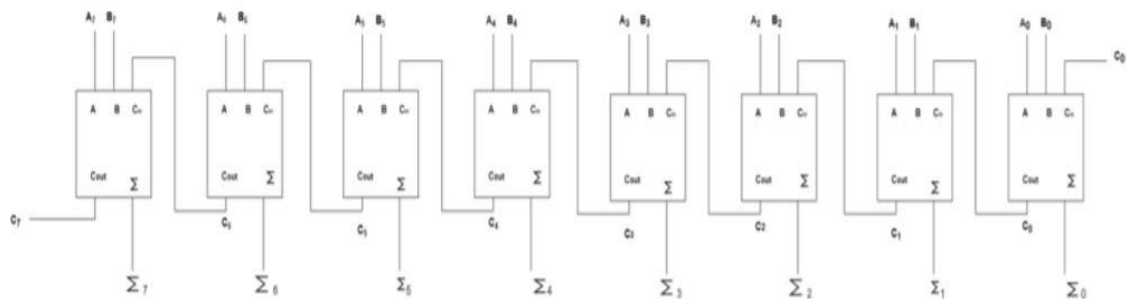


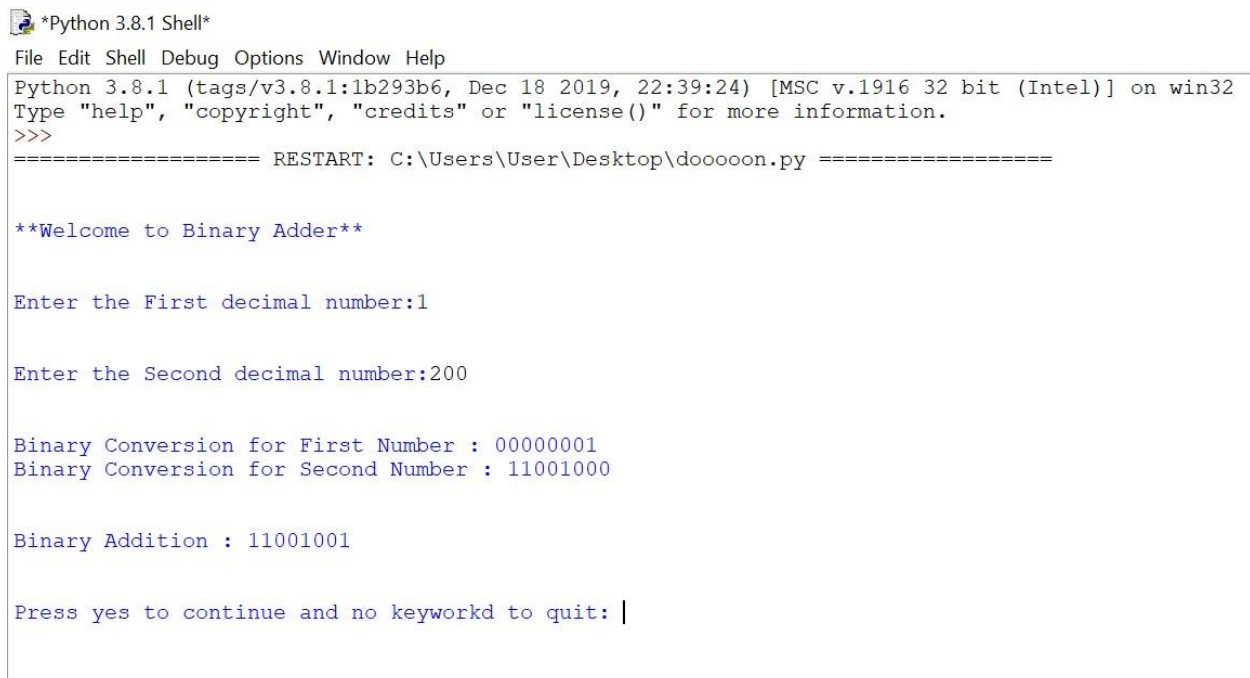
Figure 3 diagram of 8 bit adder

The picture above is the figure of 8-bit parallel adder which takes two input a and b and performs the binary addition of the two numbers and carry out the carry number from one bit to another bit and similarly in the second bit same tasks is performed where the carry is also taken in input and added and if the remainder is there then similarly from first bit to second bit, the second bit remainder goes to third bit and so on to 8 bit and the output is finally given at the end.

## 4.Data Structure

Data structure are used to storing, organizing, managing and data in easier access and efficient modifications. Data Structures allows you to organize your data in such a way that enables you to store collections of data, relate them and perform operations on them accordingly. Some of the data structure which are used in this python are append(),extend() ,insert() . while the append() method adds an item to the end of the list.Python has implicit support for Data Structures which enable you to store and access data. These structures are called List, Dictionary, Tuple and Set. (docs.python.org/3/tutorial/datastructures, n.d.)

## 5.Testing



```

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\User\Desktop\doooooon.py =====

**Welcome to Binary Adder**

Enter the First decimal number:1

Enter the Second decimal number:200

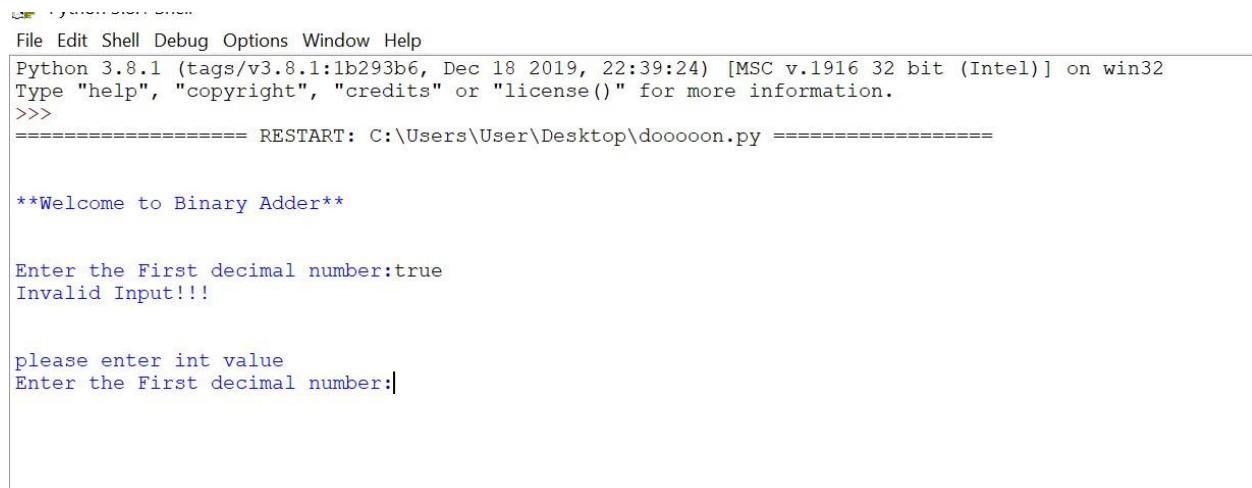
Binary Conversion for First Number : 00000001
Binary Conversion for Second Number : 11001000

Binary Addition : 11001001

Press yes to continue and no keyworkd to quit: |

```

*Figure 4testing minimum value*



```
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\User\Desktop\doooooon.py =====

**Welcome to Binary Adder**

Enter the First decimal number:true
Invalid Input!!!

please enter int value
Enter the First decimal number:|
```

*Figure 5testing boolean as input*



```
*Python 3.8.1 Shell*
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\User\Desktop\doooooon.py =====

**Welcome to Binary Adder**

Enter the First decimal number:pqr
Invalid Input!!!

please enter int value
Enter the First decimal number:|
```

*Figure 6testing random character*

```

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\User\Desktop\doooooon.py =====

**Welcome to Binary Adder**

Enter the First decimal number:20

Enter the Second decimal number:256

!!!Number invalid!!!

Please re-enter the number between 0 to 255:|

```

*Figure 7testing greater than maximum value*

```

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\User\Desktop\doooooon.py =====

**Welcome to Binary Adder**

Enter the First decimal number:40

Enter the Second decimal number:100

Binary Conversion for First Number : 00101000
Binary Conversion for Second Number : 01100100

Binary Addition : 10001100

Press yes to continue and no keyworkd to quit: yes
Thanks for using byte adder :-|
>>> |

```

*Figure 8testing rndom numbers*

## 6.Conclusion

However completing this coursework was a lot harder then it seems because we may encounter any sorts of problem at any time and in my program I encountered a lots of bugs too. But with the help of teacher guidelines I was able to complete this coursework. At the end of this coursework I realized what actually I learned from this coursework was a lot more and I also realized that all knowledge that I gain in my lecture and lab was implemented here too. To be specific I have gained quite good knowledge about the way of programming, how to use functions, conditions, string, tuples, lists and algorithm. Furthermore I have learned about using gates for condition and using that case for making flowchart. Plus I have learned about 8 – bit adder and how those bits functions and how to use gates in programming style. I feel happy about gaining such knowledge from this coursework which was not easy and I encountered a lots of error but now I would like to say that I would like to receive similar more interesting coursework in the upcoming future for improving my knowledge.

## 7.Appendix

```
print("***Welcome to the Binary Adder**")
```

```
print("\n")
```

```
input = "yes"
```



```
while input == "True" or input == "T":

    output = False

    while output == False:

        try:

            H=int(input("Please enter the first number: "))

            if H>255:

                print("The number should be in range of 0 and 255")

            else:

                output = True

        except:

            print("Please Enter a valid input!")

    output = False

while out == False:

    try:

        N= int(input("Please enter the second number: "))

        if N>255:

            print("The number must be in range of 255: ")

        else:

            out= True

    except:

        print("please input a valid number!")
```

```
print("The binary value of {}is:{}\n".format(H,binary_conversion(H)))
```

```
print("The binary value of {}is:{}\n".format(N,binary_conversion(N)))
```

```
Y=binary_conversion(H)
```

```
Z=binary_conversion(N)
```

```
sum1= binaryAdder(str(Y),str(Z))
```

```
print("The sum of {} and {} is:{}\n".format(H,N,sum1))
```

```
def binaryConversion(n):
```

```
    bit=[]
```

```
    Binary_Number=""
```

```
    counter=0
```

```
    while counter!=8:
```

```
        reminder=n%2
```

```
        bit.append(reminder)
```

```
        n=n//2
```

```
        counter=counter+1
```

```
    return bit
```

```
def reverse(bit):
```

```
    Binary=[]
```

```
Binary_Number=""
```

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

```
    Binary.append(bit[i])
```

```
    BinNumber=Bin_Number+str(bit[i])
```

```
return Bin_Number
```

```
#not gate
```

```
def not_gate(y):
```

```
    if Y==0:
```

```
        return '1'
```

```
    else:
```

```
        return '0'
```

```
#xor gate
```

```
def xor_gate(Y,Z):
```

```
    Y1 = and_gate(Y, not_gate(Z))
```

```
    Z1 = and_gate(Z, not_gate(Y))
```

```
    return int(or_gate(Y1,Z1))
```

```
#and gate
```

```
def and_gate(Y,Z):
```

```
    if (Y=='1' and Z=='1'):
```

```
    return '1'
```

```
else:
```

```
    return '0'
```

```
#or gate
```

```
def or_gate(Y,Z):
```

```
    if(Y==Z=='0'):
```

```
        return '0'
```

```
    else:
```

```
        return '1'
```

```
#byte adder
```

```
#def binaryAdder(Y,Z):
```

```
def XYZ(binaryNumber1,binaryNumber2):
```

```
#converting binarynumber to string
```

```
    str1=str(binaryNumber1)
```

```
    str2=str(binaryNumber2)
```

```
#converting to 8 digit bytes
```

```
for i in range(0,(8- len(str1) ),+1) :
```

```
    str1= '0+ str1'
```

```
for i in range(0,(8 - len(str2)),+1) :
```

```
    str2= '0+ str2'
```

```
_sum1=""
```

```
carry='0'
```

```
for i in range (7,-1,-1):
```

```
    Y,Z=str1[i],str2[i]
```

```
    _sum1= _xor(xor(X,Y),_and(_xor(Y,Z),carry))
```

```
if(carry==0):
```

```
    return _sum
```

```
else:
```

```
    print("sorry,sum is too much huge for stored in byte")
```

```
#Byte Adder('1','1')
```

```
Loop=True
```

```
while Loop==True:
```

```
    stringfirst=False
```

```
    while stringfirst==False:
```

```
        try:
```

```
            binaryNumber1=int(input("Enter the decimal number first:"))
```

```
            binaryNumber1=valid(binaryNumber1)
```

```
            print("\n")
```

```
            stringfirst=True
```

```
        except:
```

```
            print("Invalid Input!")
```

```
            print("\n")
```

```
            print("please enter int value")
```

```
stringsecond=False
```

```
while stringsecond==False:
```

```
    try:
```

```
        binaryNumber2=int(input("Enter the decimal number second:"))
```

```
        binaryNumber2=valid(binaryNumber2)
```

```
        print("\n")

        stringsecond=True

    except:

        print("Invalid Input!")

        print("\n")

        print("please enter int value")

reversefirst=binaryConversion(binaryNumber1)

reversesecond=binaryConversion(binaryNumber2)


first=reverse(reversefirst)

second=reverse(reversesecond)

XYZ(first,second)

result = XYZ(first,second)


#MainClass()

print("Binary Conversion of initial number is :",initial)

print("Binary conversion of fianl number is :",final)

print("\n")

try:

    q=int(input("Do you want to continue and Do you want to leave:"))

    if q==yes:
```

```
print()
```

```
print("continueing----")
```

```
else:
```

```
    print("Thank u for using byte adder :)")
```

```
    break
```

```
except:
```

```
    print("thank u for using byte adder:)")
```

```
    break
```



## 8. References

[docs.python.org/3/tutorial/datastructures](https://docs.python.org/3/tutorial/datastructures), n.d. s.l.:s.n.

\