



slingshot college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4051NI: Fundamentals of Computing

Assessment Weightage & Type

100% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Pratik Amatya

Group: C1

London Met ID: 19031389

College ID: NP01CP4A190024

Assignment Due Date: Monday 8 June 2020

Assignment Submission Date: Monday 8 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

1. Introduction	1
1.2. Goal and Objectives	1
1.3. Use of the application	2
1.4. Tools Used.....	2
2. Model	3
2.1. Logic Gates.....	3
2.1.1. AND Gate.....	3
2.1.2. OR Gate.....	3
2.1.3. XOR Gate.....	4
2.2. Half Adder Circuit.....	5
2.3. Full Adder Circuit	6
2.4. 8-Bit Adder Circuit.....	7
3. Algorithm	9
4. Pseudocode	12
4.1. MainProgram Module	12
4.2. NumberValidation Module	13
4.3. NumberSystemConversions Module	14
4.4. BitAdder Module	15
4.5. ReverseList Module	16
5. Flowchart.....	16
6. Data Structures	22
6.1. String	23
6.2. Integers.....	24
6.3. List.....	25

6.4. Boolean	26
7. Testing	27
7.1. Testing 1 – The user enters normal values.....	27
7.2. Testing 2 – The user enters values resulting to 9-bit operation	28
7.3. Testing 3 – Continuous Loop	29
7.4. Testing 4 – The user enters wrong data type (Exception Handling)	31
7.5. Testing 5 – The user enters value out of range	32
8. Conclusion	33
9. Bibliography	34

List of Figures

Figure 1: Symbol of AND Gate	3
Figure 2: Symbol of OR Gate	3
Figure 3: Symbol of XOR Gate.....	4
Figure 4: Half Adder Circuit	5
Figure 5: Full Adder Circuit.....	6
Figure 6: 8-Bit Adder Circuit [Starting from left to right]	7
Figure 7: Screenshot of Flowchart	21
Figure 8: Example of use of String data type	23
Figure 9: Screenshot of string data type in BitAdder.py module.....	23
Figure 10: Example of use of Int data type.....	24
Figure 11: Screenshot of Int data type in MainProgram.py module.....	24
Figure 12: Example of use of List data type	25
Figure 13: Screenshot of List data type in NumberSystemConversions.py module	25
Figure 14: Example of use of Boolean data type.....	26
Figure 15: Screenshot of Boolean Data Type in MainProgram.py module.....	26
Figure 16: Screenshot of Test 1 – The user enters normal values.....	27
Figure 17: Screenshot of Test 2 – The user enters special values.....	28
Figure 18: The user enters the normal values - Continuous Loop.....	29
Figure 19: The user enters “Y” to enter numbers again - Continuous Loop	29
Figure 20: The User is allowed to enter new values again – Continuous Loop	30
Figure 21: Screenshot of Test 4 – The user enters String Data Type	31
Figure 22: Screenshot of Test 4 – The user enters Float Data Type.....	31
Figure 23: Screenshot of Test 5 – The user enters value out of range	32

List of Tables

Table 1: Truth Table of AND Gate.....	3
Table 2: Truth Table of OR Gate	4
Table 3: Truth Table of XOR Gate	4
Table 4: Truth Table of Half Adder Circuit.....	5
Table 5: Truth Table of Full Adder Circuit.....	6
Table 6: Truth Table of 8-Bit Adder Circuit – Part I	7
Table 7: Truth Table of 8-Bit Adder Circuit – Part II	8
Table 8: The user enters normal values.....	27
Table 9: The user enters values resulting to 9-bit operation.....	28
Table 10: Continuous Loop	29
Table 11: The user enters wrong data type (Exception Handling).....	31
Table 12: The user enters value out of range.....	32

1. Introduction

This coursework was given to us in order to develop a software application which simulates the behaviour of a digital circuit performing integer addition. A model of a byte adder was designed with the combination of electronic gates.

In order to write the program, python programming language was used, and the program was written in python IDLE 3.8.1 which implements the model of the byte adder. The main function of the code is to convert decimal into binary and find the binary sum.

The program contains 5 modules. The algorithm, pseudocode of each module and flowchart of the program was also developed.

1.2. Goal and Objectives

Goal

The main goal of this project is to develop an application which performs the integer addition simulating the behaviour of the digital circuit.

Objectives

The following are the objectives of the project:

- Plenty of research to be done on topics like algorithms, flowchart, bit adder, logic gates and pseudocode.
- An algorithm to be written of the program to have proper understanding of the program.
- The pseudocode to be written of each module to code the program easier.
- The flowchart should be drawn based on the algorithm to have a clear view of the flow of the program.
- The program to be written with proper indentation and comments which gives the binary sum of two numbers taken from the user if the binary sum is less than or equals to 8-bit else appropriate message would be shown.
- Test the program to find any errors and fix to make it correct.

1.3. Use of the application

The use of this application is to find the binary sum of two decimal numbers. It allows the user to enter numbers again until the user does not want to.

1.4. Tools Used

- Python IDLE 3.8.1

It is the Integrated Development and Learning Environment for python. It is used for coding the program.

- Draw.io

It is a free online diagram editor which is used to create UML, flowcharts, entity relation diagrams, etc. It is used to make the flowchart in this report.

In this project, it is used in creating the Parallel Cascading Circuit, Flowchart.

- Logically

It is an application used to design circuits easily.

It is used to develop the half adder circuit, the full adder circuit and the 8-bit adder circuit.

- Snipping tool

It is a screenshot utility found in Microsoft Windows.

It is used to take screenshots of the testing cases and codes.

2. Model

2.1. Logic Gates

Logic Gates is an electronic circuit having one or more than one input and only one output. The relationship between the input and output is based on a certain logic. The logic gates used in this project are XOR gate, AND gate and OR gate. (Tutorialspoint, 2020)

2.1.1. AND Gate

The AND Gate gives the output as 1 only when all the inputs has value 1.

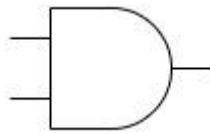


Figure 1: Symbol of AND Gate

Inputs		Output
A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth Table of AND Gate

2.1.2. OR Gate

The OR Gate gives the output as 1 when either one of the inputs are one.

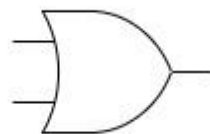


Figure 2: Symbol of OR Gate

Inputs		Output
A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Truth Table of OR Gate

2.1.3. XOR Gate

The XOR Gate gives the output as 1 only when the number of inputs having value 1 is odd.

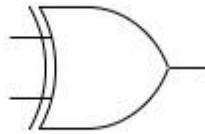


Figure 3: Symbol of XOR Gate

Inputs		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Truth Table of XOR Gate

2.2. Half Adder Circuit

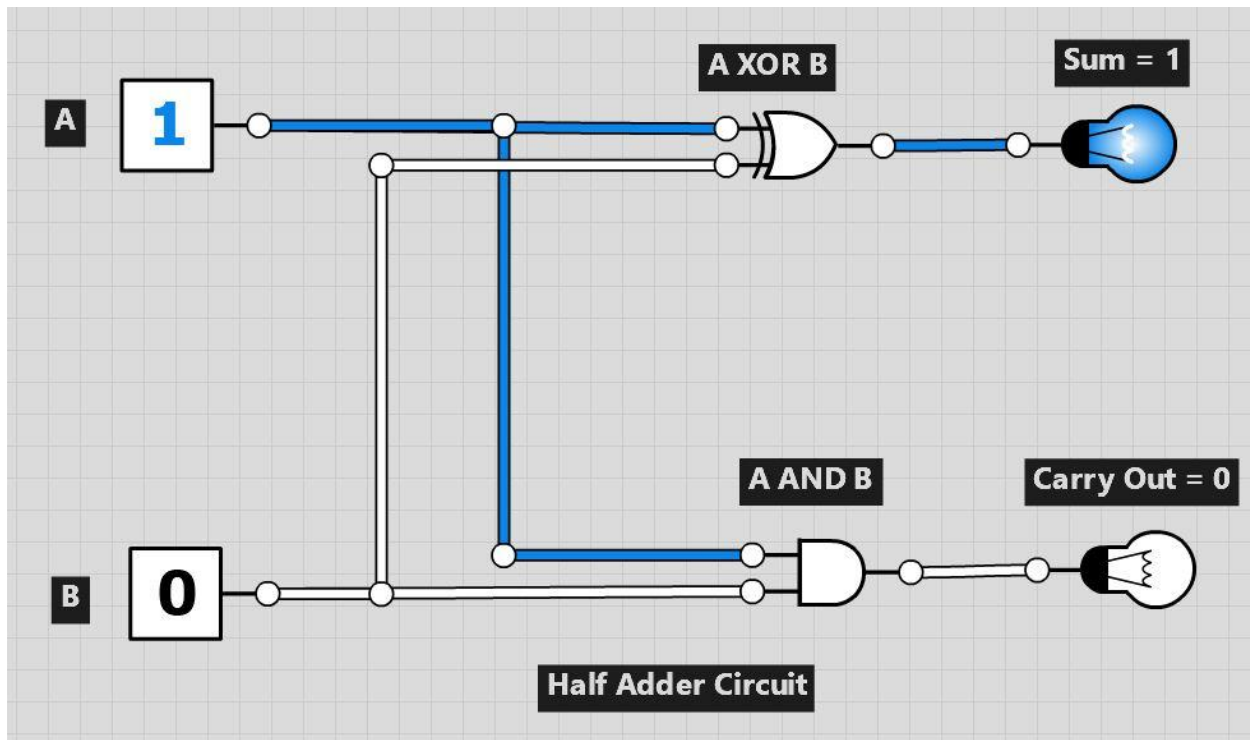


Figure 4: Half Adder Circuit

A	B	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 4: Truth Table of Half Adder Circuit

The half circuit is used for the addition of two bits. It has two inputs and gives two outputs which is Sum and Carry Out. As mentioned in the diagram above, half adder circuit uses XOR gate to determine the value of sum and AND gate to determine the value of carry out. The values of sum and carry out is dependent on the input values. All the possible values of the sum and carry out is shown in the above truth table.

2.3. Full Adder Circuit

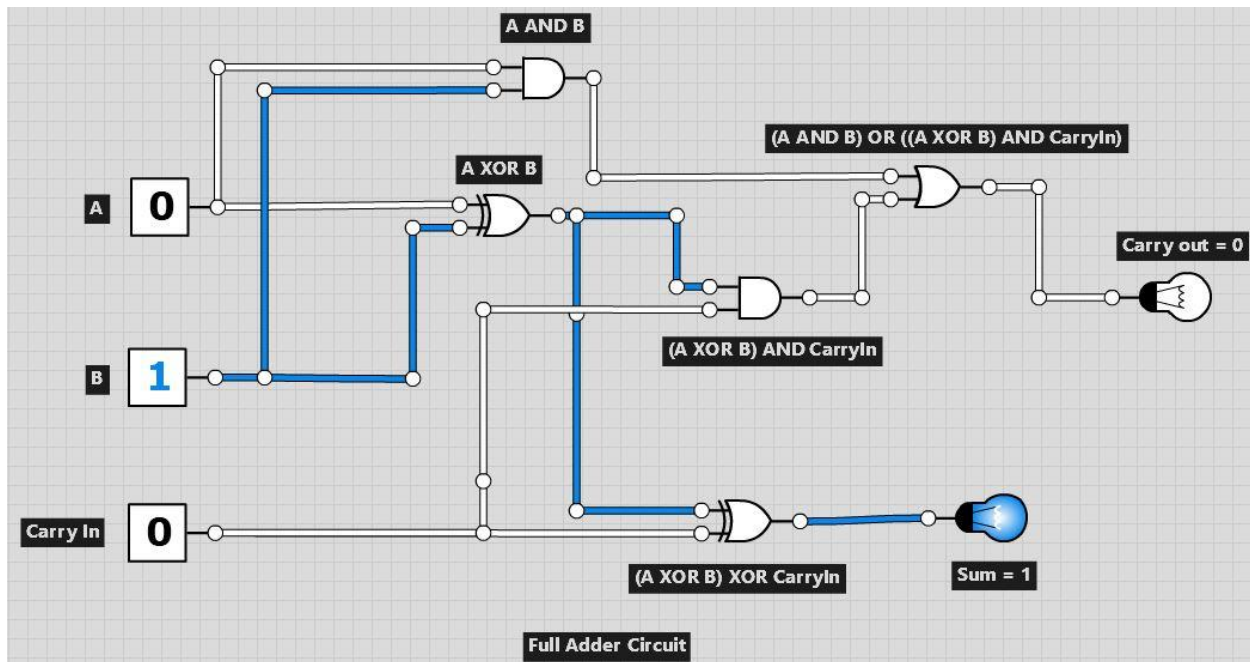


Figure 5: Full Adder Circuit

A	B	Carry in	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 5: Truth Table of Full Adder Circuit

The full circuit has three inputs and gives two outputs which is Sum and Carry Out. As mentioned in the diagram above, full adder circuit uses the combination of XOR gate to determine the value of sum. The combination used to find sum is **(A XOR B XOR Carry in)**. The combination of AND gate and XOR gate is used to determine the value of carry out. The combination used to find carry out is **((Carry in AND (A XOR B)) OR (A AND B))**.

The different outputs of sum and carry out is dependent on the values of the input values. All the possible values of the sum and carry out is shown in the above truth table.

2.4. 8-Bit Adder Circuit

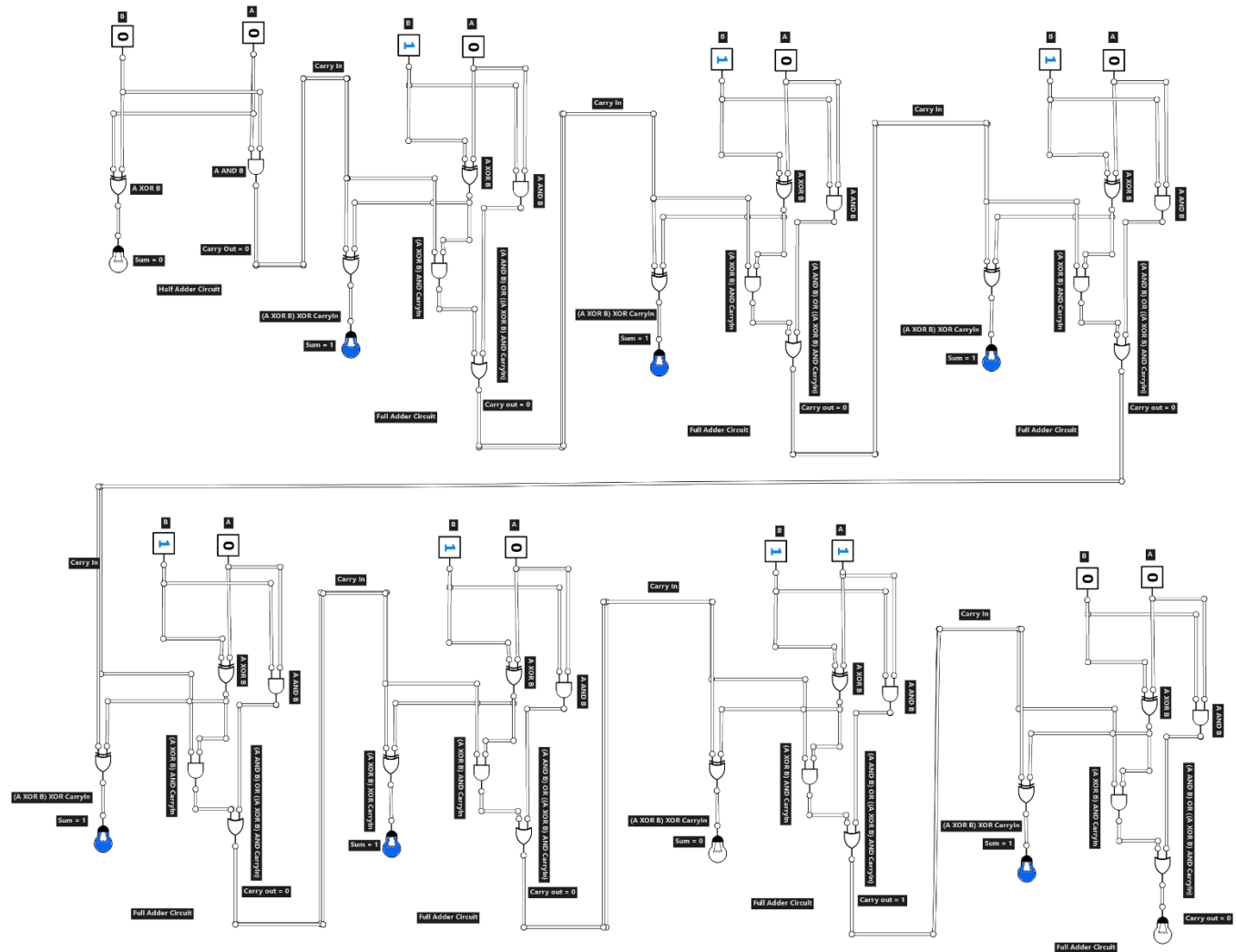


Figure 6: 8-Bit Adder Circuit [Starting from left to right]

A	B	Sum	Carry Out
0	0	0	0

Table 6: Truth Table of 8-Bit Adder Circuit – Part I

A	B	Carry in	Sum	Carry Out
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0

Table 7: Truth Table of 8-Bit Adder Circuit – Part II

8-bit Adder Circuit is used for addition of two 8-bit integers. In the project, the 8-bit adder circuit is designed by using the combination of half adder circuit and full adder circuit. The circuit above is a combination of 7 full adders and 1 half adder. The half adder provides the input i.e. carry in for the next full adder. Similarly, the n_{th} full adder provides the value of carry in for the $(n + 1)_{th}$ full adder.

The figure 6 shows the binary addition of the two decimal numbers 64 and 126. The two decimal numbers are converted to binary number systems. Additional 0 bits are added before the integer if the number is not 8-bit. Then, the digits of corresponding positions are added to the 8-bit adder circuit from the last to first digits. The last digits of two numbers are assigned to A and B of the half adder circuit. The second last digits are assigned to the next full adder circuit and so on. The sum and carry out of the last digits are found through half adder circuit. The carry out is set as the carry in for the next full adder circuit. The full adder generates the carry in for the next full adder circuit and so on until at the final full adder circuit. When the process is completed, we will get the output bits Sum and Carry Out. The sum and carry out of the first half adder in the 8-bit adder circuit is shown in the table 6. The sum and carry out of the remaining full adder in the 8-bit adder circuit is shown in the table 7.

3. Algorithm

“An algorithm is a set of instructions designed to perform a specific task. This can be a simple process, such as multiplying two numbers, or a complex operation, such as playing a compressed video file. Search engines use proprietary algorithms to display the most relevant results from their search index for specific queries.” (TechTerms, 2013)

The algorithm for the code of the coursework is given below:

- **Step 1:** Start
- **Step 2:** Assign number1 as the value of input value
- **Step 3:** If (number1 ≥ 0 and number1 ≤ 255) go to Step 4. Else go to Step 2.
- **Step 4:** Assign number2 as the value of input value
- **Step 5:** If (number2 ≥ 0 and number2 ≤ 255) go to Step 6. Else go to Step 4.
- **Step 6:** Initialize binaryList1 and binaryList2 as empty list.
- **Step 7:** If (number1 > 0) go to Step 8. Else go to Step 11.
- **Step 8:** If (number1 % 2 = 0) add 0 to the binaryList1. Else add 1 to the binaryList1.
- **Step 9:** Set value of number1 as (number1//2)
- **Step 10:** Go to Step 7.
- **Step 11:** If (number2 > 0) go to Step 12. Else go to Step 14.
- **Step 12:** If (number2 % 2 = 0) add 0 to the binaryList2. Else add 1 to the binaryList2.
- **Step 13:** Go to Step 11.
- **Step 14:** If len(binaryList1) $\neq 8$ go to Step 15. Else go to Step 20.
- **Step 15:** Set loopNumber1 as 8 – len(binaryList1)
- **Step 16:** Set i as 0
- **Step 17:** If i < loopNumber1 go to Step 18. Else go to Step 20.
- **Step 18:** Set i = i + 1 and binaryList1.append(0).
- **Step 19:** Go to Step 17.
- **Step 20:** Set reversedList1 as empty list and i as (Len(binaryList1) - 1)
- **Step 21:** If i > -1 go to Step 22. Else go to Step 24.
- **Step 22:** reversedList1.append(binaryList1[i]) and Set i = i – 1

- **Step 23:** Go to Step 21.
- **Step 24:** If $\text{len}(\text{binaryList2}) \neq 8$ go to Step 25. Else go to Step 30.
- **Step 25:** Set loopNumber2 as $8 - \text{len}(\text{binaryList2})$
- **Step 26:** Set $i = 0$
- **Step 27:** If $i < \text{loopNumber2}$ go to Step 28. Else go to Step 30.
- **Step 28:** Set $i = i + 1$ and $\text{binaryList2.append}(0)$.
- **Step 29:** Go to Step 27
- **Step 30:** Set reversedList2 as empty list and i as $(\text{Len}(\text{binaryList2}) - 1)$
- **Step 31:** If $i > -1$ go to Step 32. Else go to Step 34.
- **Step 32:** $\text{reversedList2.append}(\text{binaryList2}[i])$ and Set $i = i - 1$
- **Step 33:** Go to Step 31.
- **Step 34:** Set binaryList1 as reversedList1
- **Step 35:** Set binaryList2 as reversedList2
- **Step 36:** Set $\text{carry_in} = 0$, $\text{total_sum} = 0$, $\text{carry_out} = 0$, $\text{binarysum_list} = []$ and $i = 7$
- **Step 37:** If $i > -1$ go to Step 38. Else go to Step 47.
- **Step 38:** Set a as $\text{binaryList1}[i]$ and b as $\text{binaryList2}[i]$
- **Step 39:** If $i = 7$ go to Step 40. Else go to Step 41.
- **Step 40:** Set $\text{total_sum} = a^b$ and $\text{carry_out} = a \& b$
- **Step 41:** Set $\text{total_sum} = a^{(b \wedge \text{carry_in})}$ and $\text{carry_out} = (\text{carry_in} \& (a^b)) | (a \& b)$
- **Step 42:** Set $\text{carry_in} = \text{carry_out}$ and $\text{binarysum_list.append}(\text{total_sum})$
- **Step 43:** If $i = 0$ go to Step 44. Else go to Step 45.
- **Step 44:** Set $\text{binarysum_list.append}(\text{carry_out})$
- **Step 45:** Set $i = i - 1$
- **Step 46:** Go to Step 37.
- **Step 47:** Set $\text{reversedList3} = []$ and $i = (\text{Len}(\text{binarysum_List}) - 1)$
- **Step 48:** If $i > -1$ go to Step 49. Else go to Step 51.
- **Step 49:** Set $\text{reversedList3.append}(\text{binarysum_List}[i])$ and $i = i - 1$
- **Step 50:** Go to Step 48.
- **Step 51:** Set $\text{binarysum_list} = \text{reversedList3}$, $\text{binary_sum} = "$ " and $i = 0$
- **Step 52:** If $i < \text{len}(\text{binarysum_list})$ go to Step 53. Else go to Step 55.

- ➔ **Step 53:** Set `binary_letter = str(binarysum_list [i])`, `binary_sum = binary_sum + binary_letter` and `i = i +1`
- ➔ **Step 54:** Go to Step 52.
- ➔ **Step 55:** `binary_sum=int(binary_sum)`
- ➔ **Step 56:** If `len(str(binary_sum)) <= 8` go to Step 57. Else go to Step 58.
- ➔ **Step 57:** `print(binary_sum)`
- ➔ **Step 58:** `print (Error Message)`
- ➔ **Step 59:** Assign `asking_user` as the value of input value
- ➔ **Step 60:** If `asking_user = "y"` or `asking_user = "Y"` go to Step 2. Else if `asking_user = "n"` or `asking_user = "N"` go to Step 61. Else go to Step 59.
- ➔ **Step 61:** End

4. Pseudocode

“Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program’s flow but excludes underlying details. System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly.” (The Economic Times, 2020)

4.1. MainProgram Module

IMPORT NumberValidation **AS** checker

IMPORT NumberSystemConversions **AS** conversion

IMPORT BitAdder

def main_function():

 continuation=True

WHILE continuation = True:

LET number1=checker.validate (1)

LET number2= checker.validate (2)

LET firstActualBinaryConversion= conversion.decimal_to_binary(number1)

LET secondActualBinaryConversion=
 conversion.decimal_to_binary(number2)

LET binary_sum= BitAdder.bit_add (firstActualBinaryConversion,
 secondActualBinaryConversion)

IF LENGTH(binary_sum **TO STRING**) <= 8:

DISPLAY ("-----Result-----")

DISPLAY ("The Binary Sum
 of", number1, "and", number2, "is", binary_sum)

ELSE:

PRINT ("The Binary Sum exceeds 8 bit. Cannot be performed.Please try
 again.")

```
LET askingContinuation=True
WHILE askingContinuation = True:
    LET asking_user = INPUT("Do want to enter again? (Y/N) ")
    IF asking_user="n" OR asking_user="N":
        DISPLAY ("---Thank You for using the application---")
        LET continuation = False
        BREAK
    ELSE IF asking_user="y" OR asking_user="Y":
        BREAK
    ELSE:
        DISPLAY ("Please enter Y or N only.\n")
```

4.2. NumberValidation Module

def validate(numberPosition):

```
    LET correctNumberEntered= False
    WHILE correctNumberEntered = False:
        TRY:
            IF numberPosition = 1:
                number = INPUT ("Enter the first number in decimal number
                system: ") TO INT
            ELSE:
                number = INPUT ("Enter the second number in decimal
                number system: ") TO INT
            IF number>=0 AND number<256:
                RETURN number
                BREAK
            ELSE IF number<0:
                DISPLAY ("Please enter positive numbers only")
```

CONTINUE

ELSE IF number>255:

DISPLAY ("Please enter numbers between 0 and 255 only")

EXCEPT:

DISPLAY ("Please enter whole numbers only")

4.3. NumberSystemConversions Module

IMPORT ReverseList **AS** reverse

def decimal_to_binary(firstNumVariable):

LET binaryList = []

WHILE firstNumVariable >0:

IF firstNumVariable % 2 = 0:

 binaryList.**append** (0)

ELSE:

 binaryList.**append** (1)

 firstNumVariable = firstNumVariable // 2

IF LEN (binaryList) **NOT EQUALS** 8:

 loopNumber = 8 – **LEN**(binaryList)

For i In Range (loopNumber):

 binaryList.**append** (0)

 binaryList=reverse.reverse_list(binaryList)

RETURN binaryList

4.4. BitAdder Module

IMPORT Operators **AS** operator

IMPORT ReverseList **AS** reverse

def bit_add(list1,list2):

LET carry_in = 0

LET total_sum = 0

LET binarysum_list = []

LET carry_out = 0

FOR i in Range (**START** 7, **STOP** -1, **STEP** -1):

LET a = list1[i]

LET b= list2[i]

IF i = 7:

LET total_sum = a **XOR** b

LET carry_out = a **AND** b

ELSE:

LET total_sum = a **XOR** (b **XOR** carry_in)

LET carry_out = (carry_in **AND** (a **XOR** b)) **OR**
 (a **AND** b)

LET carry_in = carry_out

LET binarysum_list.append(total_sum)

IF i = 0:

 binarysum_list.append(carry_out)

LET binarysum_list = reverse. reverse_list(binarysum_list)

LET binary_sum=""

For binaryDigit **In** binarysum_list :

LET binary_letter = binaryDigit **TO** **STRING**

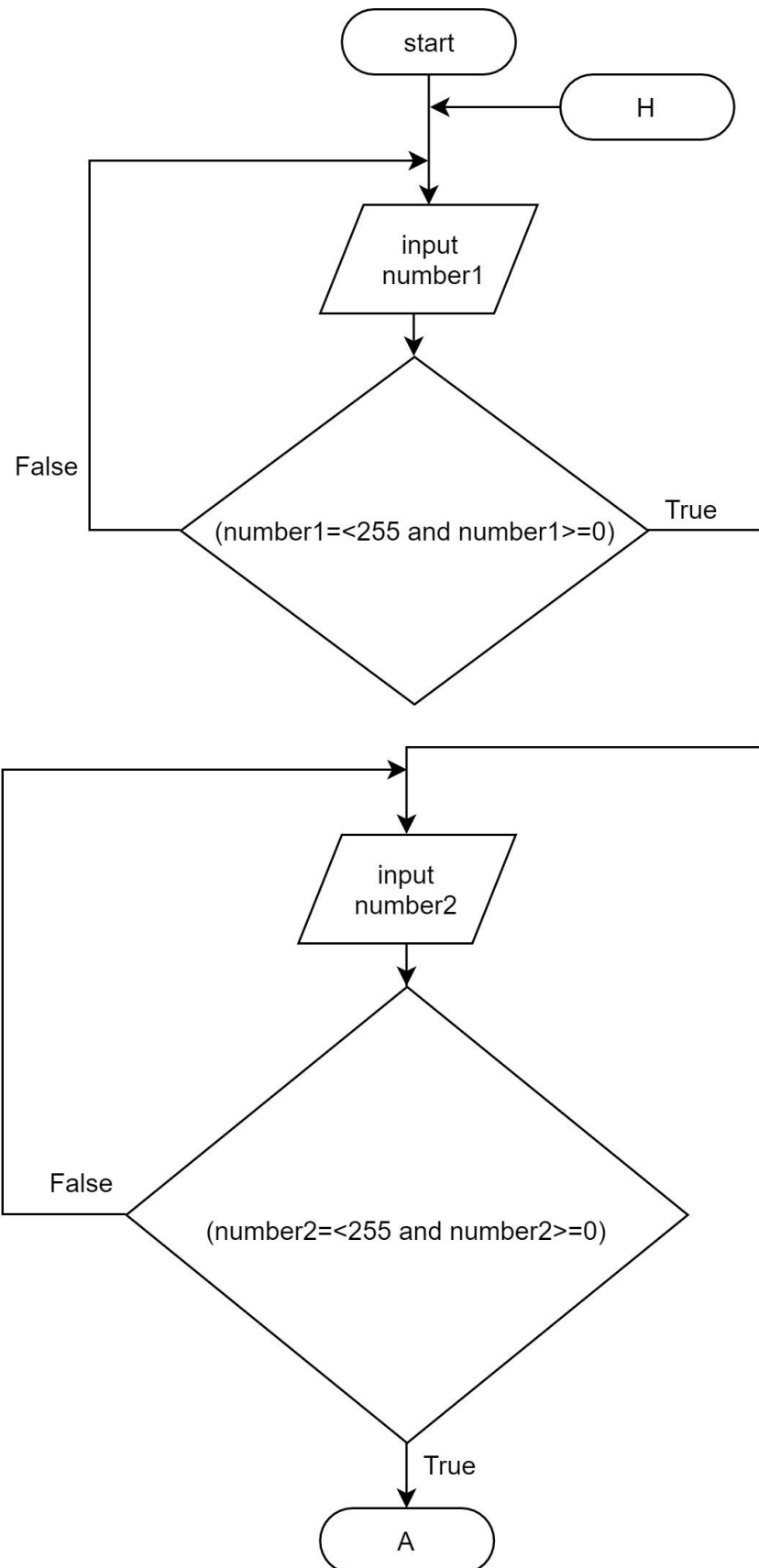
```
    LET binary_sum = binary_sum + binary_letter  
    LET binary_sum = binary_sum TO INT  
    RETURN binary_sum
```

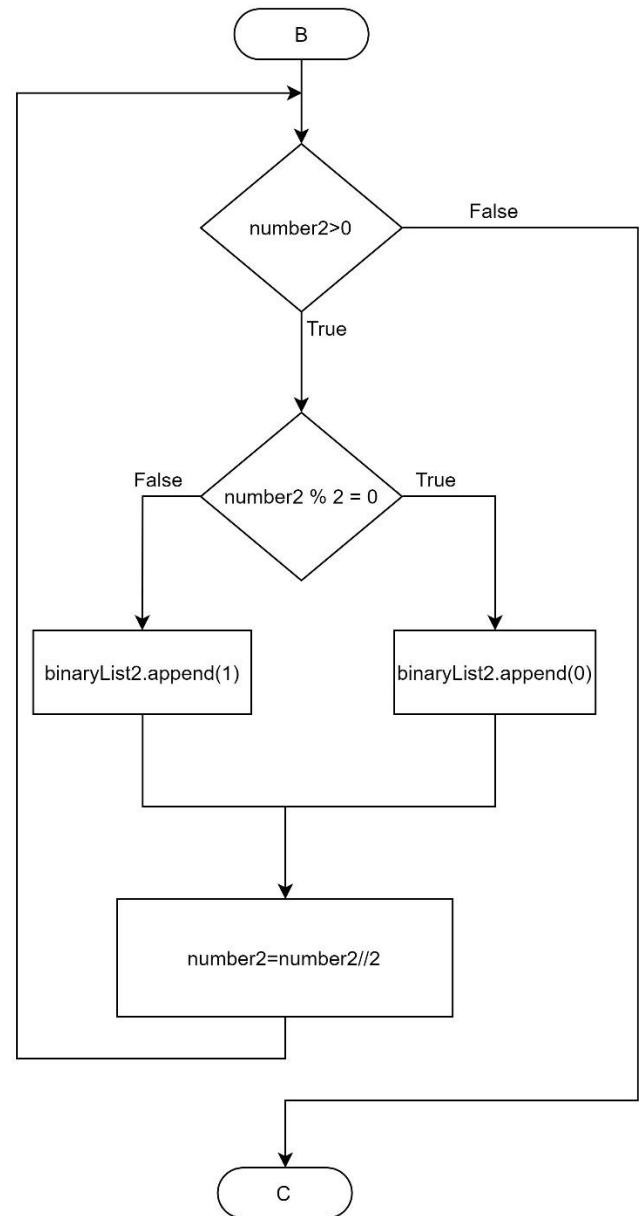
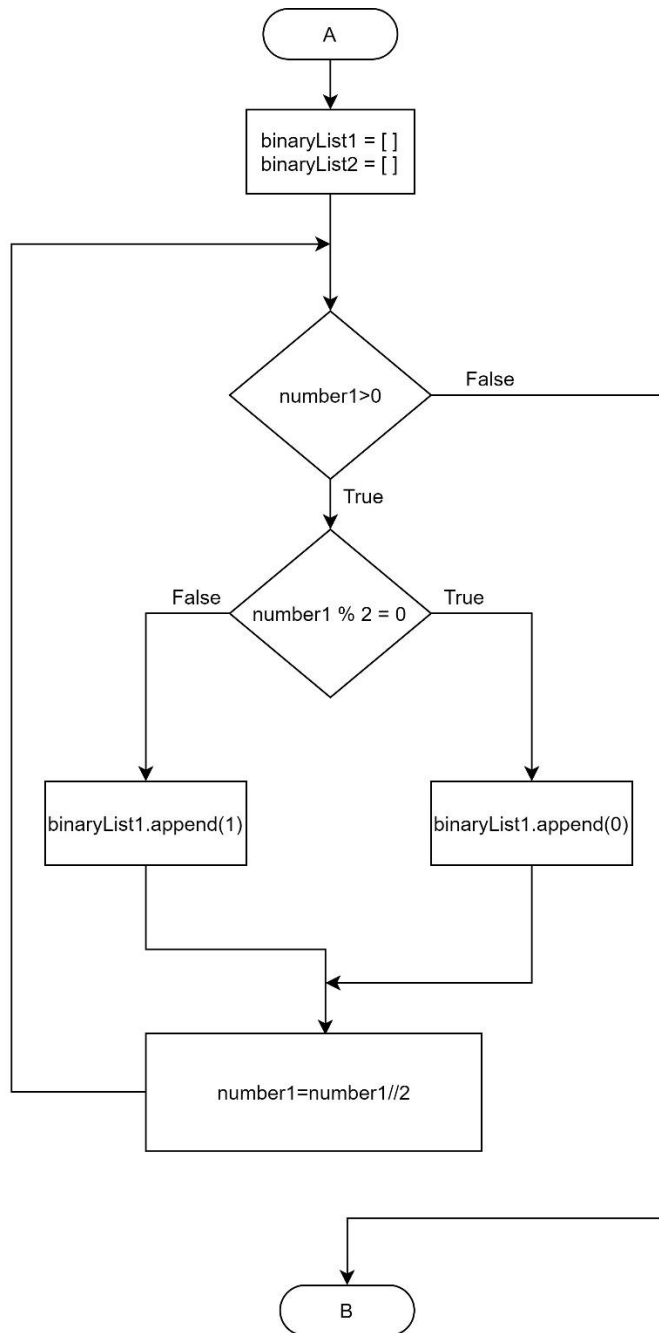
4.5. ReverseList Module

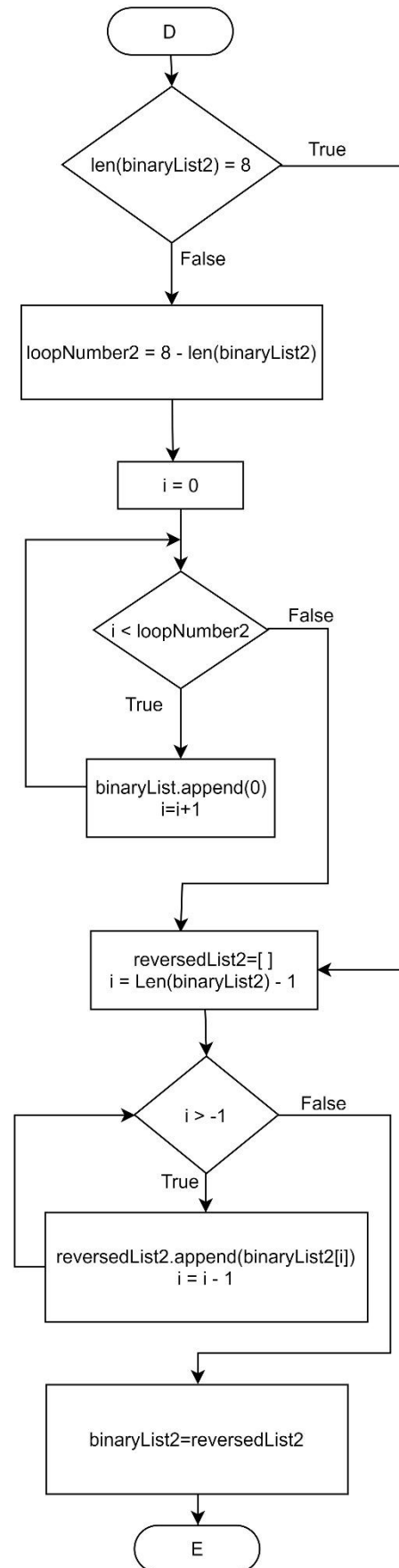
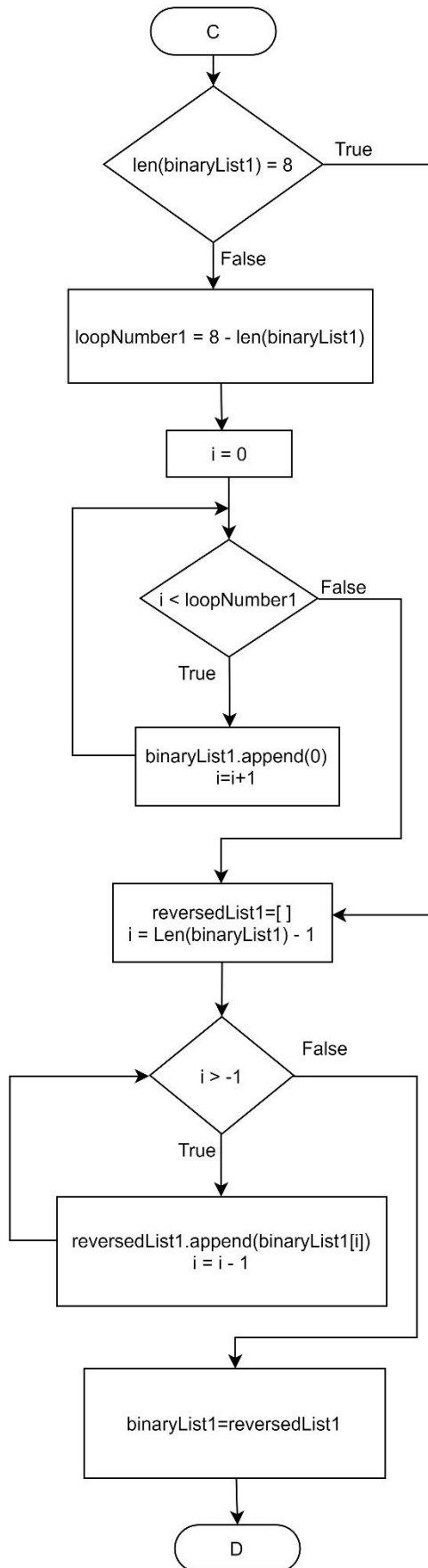
```
def reverse_list(original_list):  
    LET reversedList = [ ]  
  
    FOR i in Range (START (LENGTH(original_list)-1), STOP -1, STEP -1):  
        LET reversedList.append(original_list[ i ]) append  
    RETURN reversedList
```

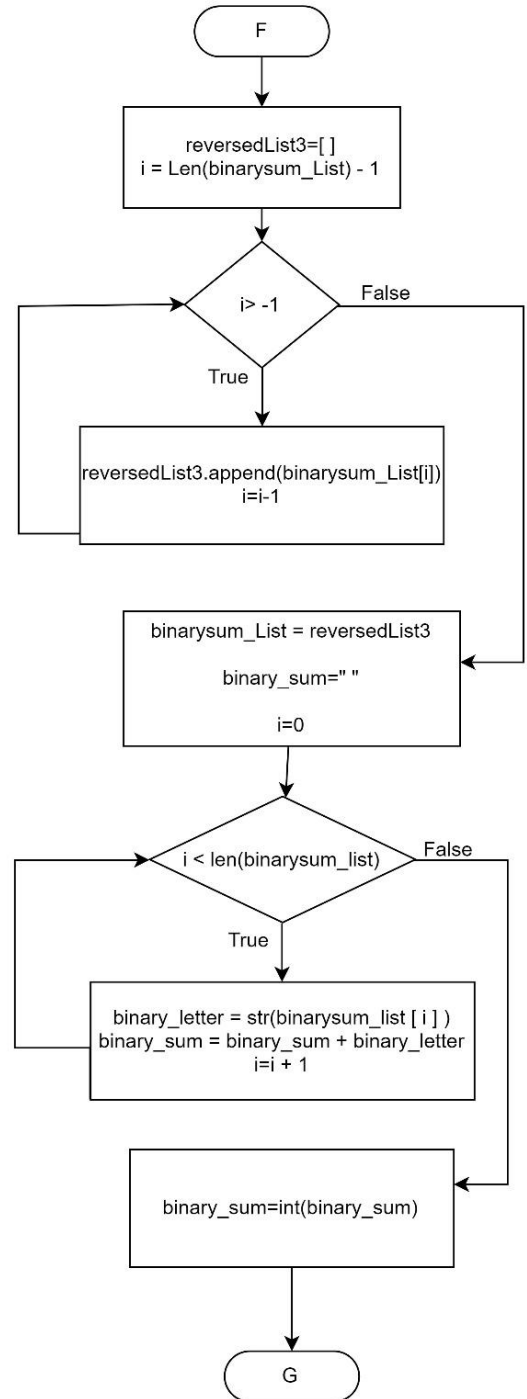
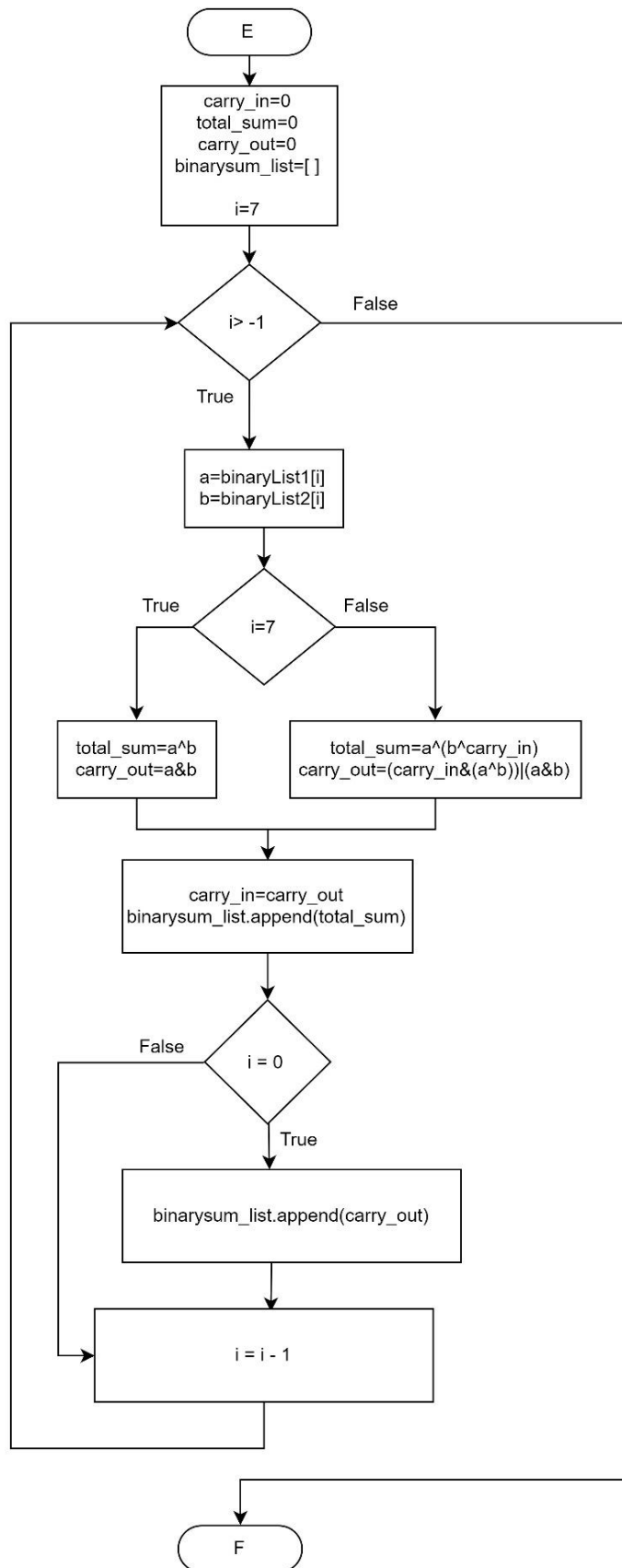
5. Flowchart

“A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes. Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.” (Visual Paradigm, 2020)









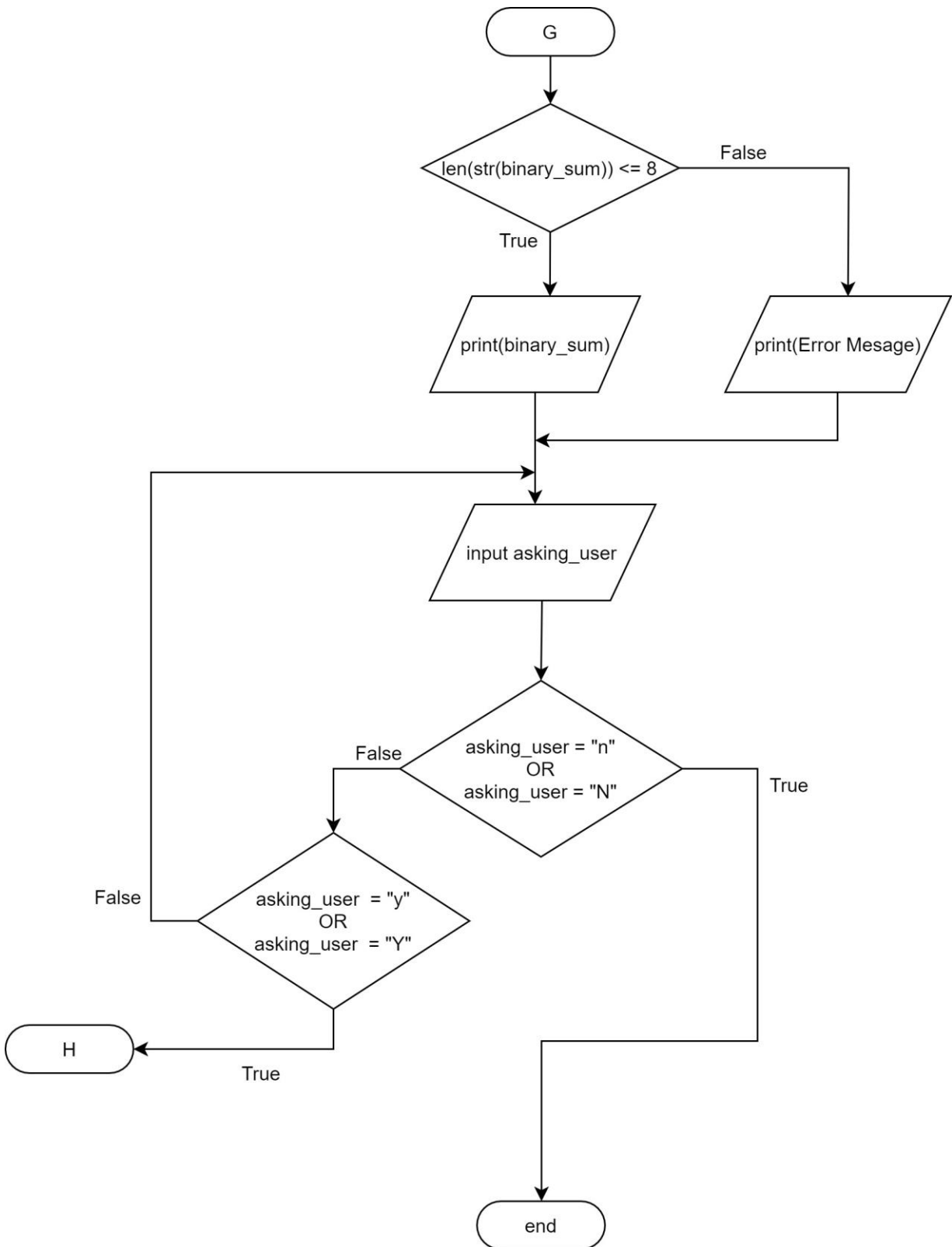


Figure 7: Screenshot of Flowchart

6. Data Structures

“Data structures are a way of organizing and storing data so that they can be accessed and worked with efficiently. They define the relationship between the data, and the operations that can be performed on the data. There are many various kinds of data structures defined that make it easier for the data scientists and the computer engineers, alike to concentrate on the main picture of solving larger problems rather than getting lost in the details of data description and access.” (Jaiswal, 2017/12/08)

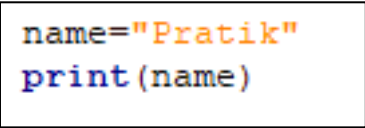
The commonly used data types in the python programming are String, Integer, Float and Boolean whereas the commonly used data structure in python programming are lists, tuples, sets and dictionaries. The data types and structures used in this program are listed below:

6.1. String

“A string object is one of the sequence data types in Python. It is an immutable sequence of Unicode characters. Strings are objects of Python's built-in class 'str'. String literals are written by enclosing a sequence of characters in single quotes ('hello'), double quotes ("hello") or triple quotes ('''hello''' or """"hello""").” (TutorialsTeacher, 2020)

Example of use of String data type in python is:

- Here, the “name” variable is used to store string data type.

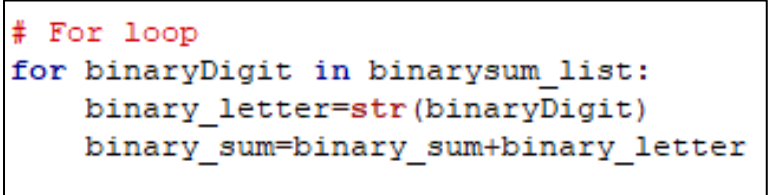


```
name="Pratik"  
print(name)
```

Figure 8: Example of use of String data type

Use of string data type in BitAdder.py module:

- Here, the list item in the “binarysum_list” list is converted to string data type and stored in “binary_letter” variable.



```
# For loop  
for binaryDigit in binarysum_list:  
    binary_letter=str(binaryDigit)  
    binary_sum=binary_sum+binary_letter
```

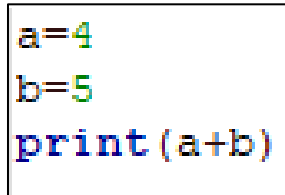
Figure 9: Screenshot of string data type in BitAdder.py module

6.2. Integers

“This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.” (GeeksforGeeks, 2020)

Example of use of Int data type in python is:

- Here, the “a” and “b” variable is used to store int data type. And the sum of the two values is printed.

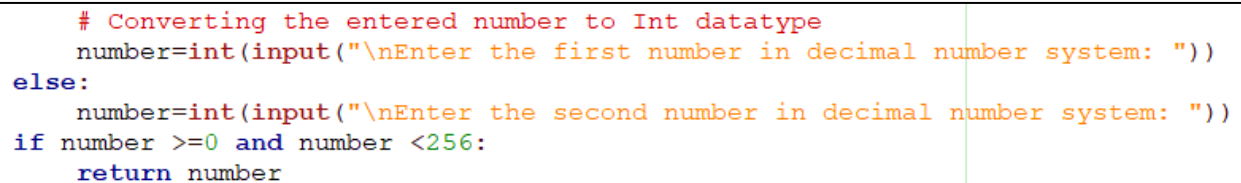


```
a=4
b=5
print(a+b)
```

Figure 10: Example of use of Int data type

Use of Int data type in MainProgram module:

- The number entered by user is taken as Int data type.



```
# Converting the entered number to Int datatype
number=int(input("\nEnter the first number in decimal number system: "))
else:
    number=int(input("\nEnter the second number in decimal number system: "))
if number >=0 and number <256:
    return number
```

Figure 11: Screenshot of Int data type in MainProgram.py module

6.3. List

“Lists are just like the arrays, declared in other languages. Lists need not be homogeneous always which makes it the most powerful tool in Python. A single list may contain Data Types like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation. List in Python are ordered and have a definite count. The elements in a list are indexed according to a definite sequence and the indexing of a list is done with 0 being the first index. Each element in the list has its definite place in the list, which allows duplicating of elements in the list, with each element having its own distinct place and credibility. It is represented by list class.” (GeeksforGeeks, 2020)

Example of use of List data type in python is:

- Here, the variable “namesOfStudent” stores the list data type.

```
namesOfStudent=["Pratik","Shyam","Hari"]  
print(namesOfStudent)
```

Figure 12: Example of use of List data type

Use of List data type in NumberSystemConversions.py module:

- The list is used to store the remainder of division of Int variable “firstNumVariable” divided by 2. The number 0 is appended to the list if the remainder of “firstNumVariable” variable divided by 2 is 0 and 1 if it is not 0.

```
binaryList=[]  
# While Loop  
while firstNumVariable>0:  
    # Adding 0 to the binaryList if the remainder when divided by 2 is 0  
    if firstNumVariable%2==0:  
        binaryList.append(0)  
    # Adding 1 to the binaryList if the remainder when divided by 2 is not 0  
    else:  
        binaryList.append(1)  
    firstNumVariable=firstNumVariable//2
```

Figure 13: Screenshot of List data type in NumberSystemConversions.py module

6.4. Boolean

“Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.” (GeeksforGeeks, 2020)

Example of use of Boolean data type in python is:

- Here, evenNumber is set as boolean value, True when remainder of number divided by 2 is 0.

```
number=2
if number%2 == 0:
    evenNumber=True
if evenNumber == True:
    print("Even Number")
```

Figure 14: Example of use of Boolean data type

Use of Boolean data type in MainProgram.py module:

- The Boolean data type is used to continue the loop while the Boolean value of variable “correctNumberEntered” is False.

```
correctNumberEntered=False
while correctNumberEntered == False:
```

Figure 15: Screenshot of Boolean Data Type in MainProgram.py module

7. Testing

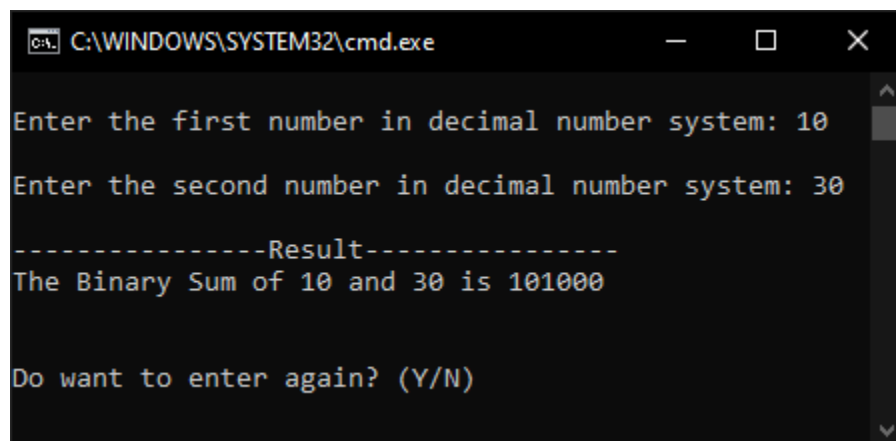
Testing is a process of executing a program with the aim of finding error. The program should be tested properly to be error free and perform properly. (GeeksforGeeks & Pankaj Patel, 2020)

7.1. Testing 1 – The user enters normal values

Test No:	1
Action:	The user enters 10 and 30
Expected Result:	The correct binary sum of the entered numbers not exceeding 8-bit would be displayed.
Actual Result:	The correct binary sum of the entered number not exceeding 8-bit was displayed.
Conclusion:	The test is successful.

Table 8: The user enters normal values

Output Result:



```

C:\WINDOWS\SYSTEM32\cmd.exe

Enter the first number in decimal number system: 10
Enter the second number in decimal number system: 30

-----Result-----
The Binary Sum of 10 and 30 is 101000

Do want to enter again? (Y/N)
  
```

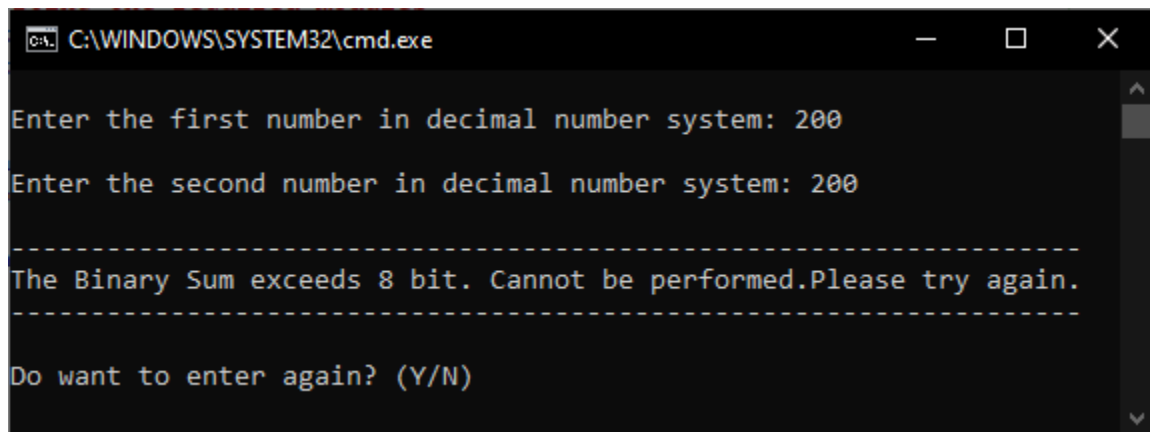
Figure 16: Screenshot of Test 1 – The user enters normal values

7.2. Testing 2 – The user enters values resulting to 9-bit operation

Test No:	2
Action:	The user enters 200 and 200.
Expected Result:	The appropriate message saying the binary sum exceeds 8-bit would be displayed as the binary sum of the entered numbers exceeds 8-bit.
Actual Result:	The appropriate message saying the binary sum exceeds 8-bit is displayed as the binary sum of the entered numbers exceeds 8-bit and becomes 9-bit operation.
Conclusion:	The test is successful.

Table 9: The user enters values resulting to 9-bit operation

Output Result:



```

C:\WINDOWS\SYSTEM32\cmd.exe

Enter the first number in decimal number system: 200
Enter the second number in decimal number system: 200

-----
The Binary Sum exceeds 8 bit. Cannot be performed. Please try again.
-----

Do want to enter again? (Y/N)

```

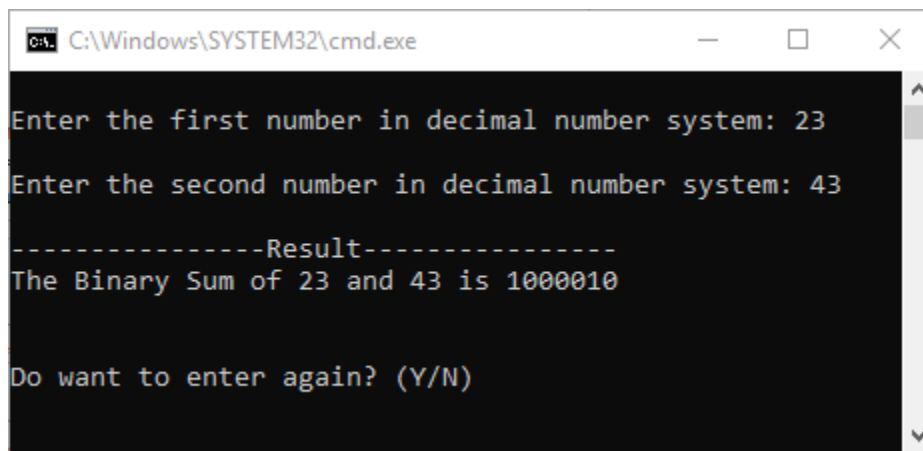
Figure 17: Screenshot of Test 2 – The user enters special values

7.3. Testing 3 – Continuous Loop

Test No:	3
Action:	<ul style="list-style-type: none"> The user enters 23 and 43. The user enters “Y” to enter the numbers again.
Expected Result:	The user would be allowed to enter different numbers again.
Actual Result:	The user was allowed to enter different numbers again.
Conclusion:	The test is successful.

Table 10: Continuous Loop

Output Result:



```

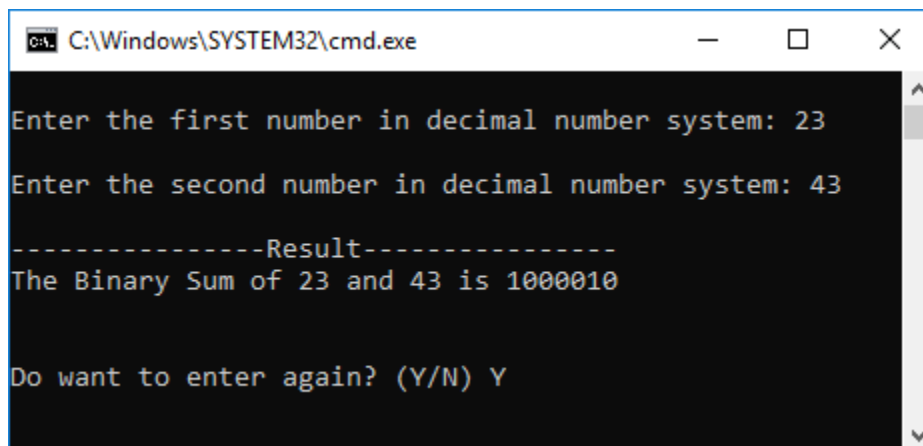
C:\Windows\SYSTEM32\cmd.exe

Enter the first number in decimal number system: 23
Enter the second number in decimal number system: 43
-----Result-----
The Binary Sum of 23 and 43 is 1000010

Do want to enter again? (Y/N)

```

Figure 18: The user enters the normal values - Continuous Loop



```

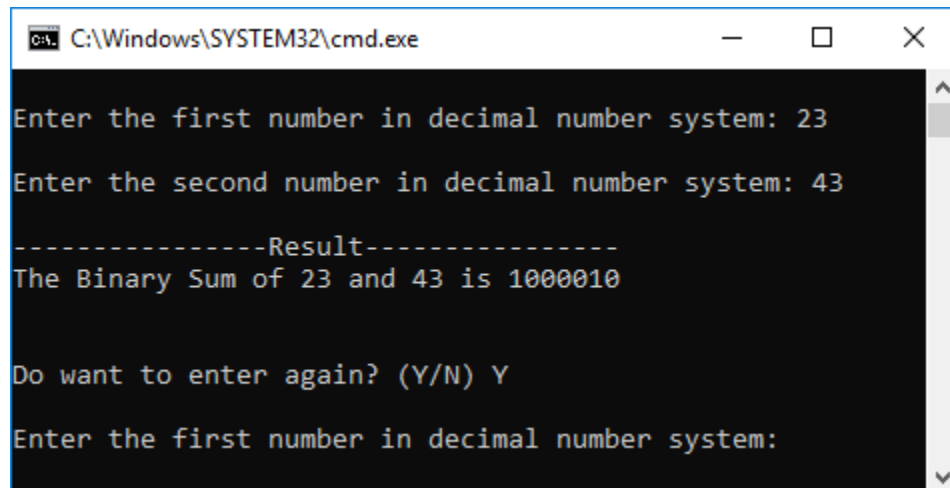
C:\Windows\SYSTEM32\cmd.exe

Enter the first number in decimal number system: 23
Enter the second number in decimal number system: 43
-----Result-----
The Binary Sum of 23 and 43 is 1000010

Do want to enter again? (Y/N) Y

```

Figure 19: The user enters “Y” to enter numbers again - Continuous Loop



```
C:\Windows\SYSTEM32\cmd.exe

Enter the first number in decimal number system: 23
Enter the second number in decimal number system: 43
-----Result-----
The Binary Sum of 23 and 43 is 1000010

Do want to enter again? (Y/N) Y
Enter the first number in decimal number system:
```

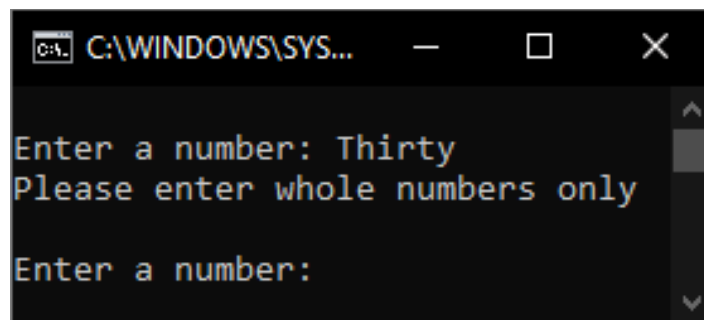
Figure 20: The User is allowed to enter new values again – Continuous Loop

7.4. Testing 4 – The user enters wrong data type (Exception Handling)

Test No:	4
Action:	<ul style="list-style-type: none"> The user enters Thirty which is a String data type The user enters 3.2 which is a float data type
Expected Result:	The appropriate message would be displayed when wrong data type is entered, and the user is allowed to enter a new value again.
Actual Result:	The appropriate message was displayed when wrong data type was entered, and the user was allowed to enter a new value again.
Conclusion:	The test is successful.

Table 11: The user enters wrong data type (Exception Handling)

Output Result:

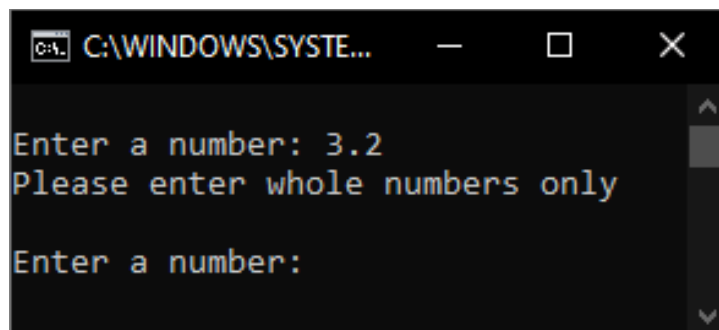


```

C:\WINDOWS\SYS...
Enter a number: Thirty
Please enter whole numbers only
Enter a number:

```

Figure 21: Screenshot of Test 4 – The user enters String Data Type



```

C:\WINDOWS\SYSTE...
Enter a number: 3.2
Please enter whole numbers only
Enter a number:

```

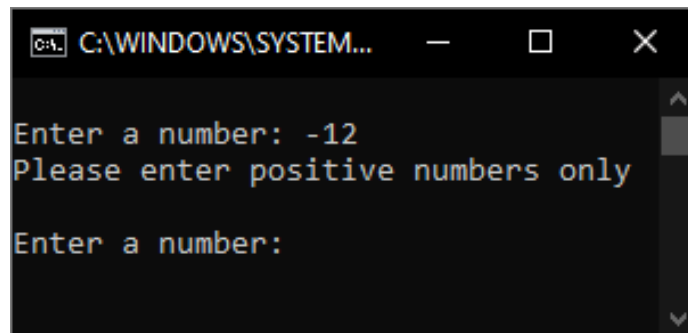
Figure 22: Screenshot of Test 4 – The user enters Float Data Type

7.5. Testing 5 – The user enters value out of range

Test No:	5
Action:	<ul style="list-style-type: none"> The user enters -12 The user enters 300
Expected Result:	The appropriate message would be displayed when the value entered by the user is out of range and the user is allowed to enter a new value.
Actual Result:	The appropriate message was be displayed when the values which are out of range was entered and the user was allowed to enter a new value.
Conclusion:	The test is successful.

Table 12: The user enters value out of range

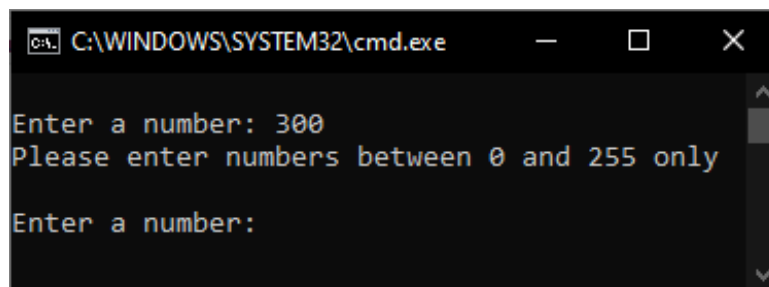
Output Result:



```

C:\WINDOWS\SYSTEM...
Enter a number: -12
Please enter positive numbers only
Enter a number:

```



```

C:\WINDOWS\SYSTEM32\cmd.exe
Enter a number: 300
Please enter numbers between 0 and 255 only
Enter a number:

```

Figure 23: Screenshot of Test 5 – The user enters value out of range

8. Conclusion

In the end, the program was coded with the implementation of logic gates. A lot research was done to finish the projects. Much of the time was spend on researching on the topics like algorithms, flowchart, bit adder, logic gates and pseudocode. Many problems were faced. To fix the issues, the lectures slides and videos uploaded by our tutor were went through. Many researches were done on the python programming from the ground level.

Many issues were faced while developing the flowchart, algorithm and the code. But doing the project helped to develop a better understanding on the concepts of python programming and improvements in writing skills. The concept of bit adder, parallel adder circuit, half adder and full adder was hard to grasp. But it was done going through the slides and study materials provided by our tutors. The feedback from our tutor through many emails helped to improve the project and get a better understanding.

The algorithm was written and then the flowchart and the pseudocode were developed. This helped to complete the code tremendously. The research helped in knowing the different data types and data structures along with their implementation in the program. The program was tested to find any issues and fix it.

The tasks were challenging and difficult having no background in python programming. With the determination and constant efforts, the project was completed and submitted on time. With the completion of the project, the concepts of python programming, bit adder, half adder circuit, full adder, logic gates, bitwise operators of python and data types and structures was learned.

9. Bibliography

GeeksforGeeks. (2020) *Python Data Types - GeeksforGeeks* [Online]. Available from: <https://www.geeksforgeeks.org/python-data-types/> [Accessed 3 May 2020].

GeeksforGeeks. (2020) *Python Data Types - GeeksforGeeks* [Online]. Available from: <https://www.geeksforgeeks.org/python-data-types/#boolean> [Accessed 5 May 2020].

GeeksforGeeks & Pankaj Patel. (2020) *Types of Software Testing - GeeksforGeeks* [Online]. Available from: <https://www.geeksforgeeks.org/types-software-testing/> [Accessed 27 April 2020].

Jaiswal, S. (2017/12/08) *Data Structures_ Python Tutorial - DataCamp* [Online]. Available from: <https://www.datacamp.com/community/tutorials/data-structures-python> [Accessed 4 May 2020].

PythonForBeginners. (2020) *Booleans, True or False in Python* [Online]. Available from: <https://www.pythonforbeginners.com/basics/boolean> [Accessed 4 May 2020].

TechTerms. (2013) *Algorithm Definition* [Online]. Available from: <https://techterms.com/definition/algorithm> [Accessed 8 May 2020].

The Economic Times. (2020) *What is Pseudocode_ Definition of Pseudocode, Pseudocode Meaning - The Economic Times* [Online]. Available from: <https://economictimes.indiatimes.com/definition/pseudocode> [Accessed 26 April 2020].

Tutorialspoint. (2020) *Logic Gates - Tutorialspoint* [Online]. Available from: https://www.tutorialspoint.com/computer_logical_organization/logic_gates.htm [Accessed 7 May 2020].

TutorialsTeacher. (2020) *Python string* [Online]. Available from: <https://www.tutorialsteacher.com/python/python-string> [Accessed 4 May 2020].

Visual Paradigm. (2020) *Flowchart Tutorial (with Symbols, Guide and Examples)* [Online]. Available from: <https://www.visual-paradigm.com/tutorials/flowchart-tutorial/> [Accessed 8 May 2020].