LONDON METROPOLITAN UNIVERSITY

islington 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: Karsang Gurung

Group: L1N6

London Met ID: 19031333

College ID: NP01NT4A190138

Assignment Due Date: 8th June, 2020

Assignment Submission Date: 8th 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

List of FIgures

List of Tables

1. Introduction

Python is one of the most commonly used programming language across the globe, which is significant, object-oriented and deciphered. The high-level data structures used in python makes it one of the best programming language for application development. The syntax in Python is much easier and simpler than that of other programming languages, keeping the cost of maintenance very low. (Anon., 2020)

Figure 1: Python logo

The main objectives of this coursework is to create a model of byte adder using different electronic gates. The sole purpose of this coursework is to be familiar with Python programming language and using the logical gates. This coursework also helps us to learn how to deal with exception handling interacting with flow charts, circuit diagrams and create a basic byte adder.

2. Model

The model of byte adder is made up of 8-bit adder. And the 8-bit adder contains different 8 electronic logic gates such as OR, XOR and AND gates. The model of the byte adder is briefly described below:

2.1 8-bit adder:

An 8-bit adder or binary adder is a digital circuit, which is used for arithmetic operations in computers like adding two 8-bit integers. It uses different logical gates to find the sum of the binary numbers. It works on 8-bit adder theory i.e. it adds numbers digit by digit.

2.2 Logic Gates:

The electronic circuit, which manages one or more input signals to produce an output, is called logic gate. The basic logic gates that are used in computer are briefly described below with truth table. The truth table is the mathematical table which shows the possible values of variables with possible results.

i. AND Gate: The AND gate portrays the logical multiplication in a circuit. It has only 1 output, but might have 2 or more inputs. Here, if both the inputs are TRUE, then the output will be TRUE. If any one of the input is FALSE then the output will be FALSE. The operation symbol of AND gate is shown by dot '.' (Subba, 2020)

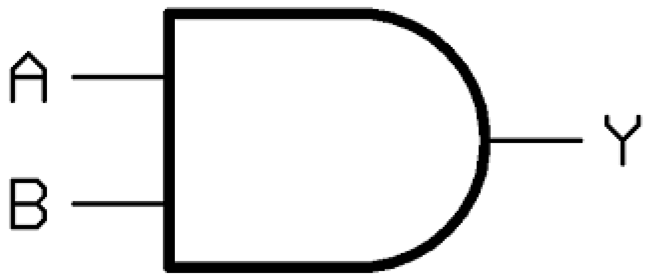

Figure 2: AND Gate

The truth table for AND Gate is given below:

Input X	Input Y	Output
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth table for AND Gate

ii. OR Gate: The OR Gate portrays the logical addition in a circuit. Similar to the AND Gate, it has only one output, but can have 2 or more inputs. If either or both the input is TRUE, the output will be TRUE. When all the inputs are FALSE, the output will be false. The operation symbol of OR gate is '+' (Subba, 2020)

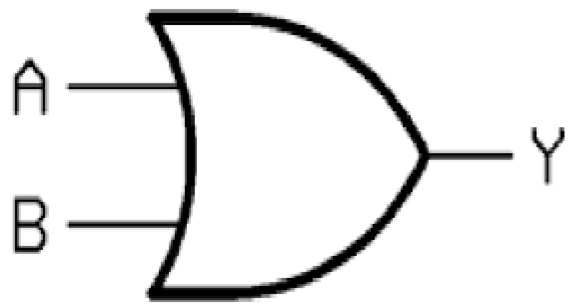

Figure 3: OR Gate

The truth table for OR Gate is given below:

Input X	Input Y	Output
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Truth table for OR Gate

iii. XOR Gate: the XOR stands for exclusive OR Gate. This operator gives the output TRUE if only the inputs are TRUE and FALSE if both the inputs are either TRUE or FALSE. The operation symbol for XOR Gate is encircled '+' (Subba, 2020)

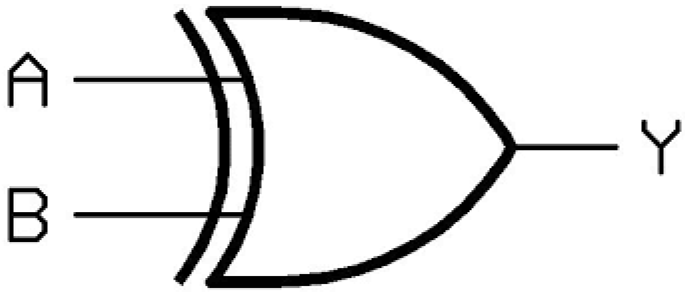

Figure 4: XOR Gate

The truth table for XOR Gate is given below:

Input X	Input Y	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Truth table for XOR Gate

2.3 Operation carrying out 8-bit adder circuit

The model created is of an 8-bit adder circuit. It takes two binary numbers as input and find out their sums as output. Here, the two inputs are input X and input Y, respectively. Sum denotes the addition of the input and operation denotes the process. Input carrier here is denoted by Cin, which is either 1 or 0. Similarly, output carrier is denoted by Cout. When the circuit is run here, the operation of addition begins with carry, Transfer X and increment X. in the same way, carry to be output, carry operation goes on.

INPUT			OUTPUT		OPERATION
A	B	Cin	Cout	Sum	
0	0	0	0	0	Add
0	0	1	0	1	Add with Carry
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	Transfer X
1	0	1	1	0	Increment Y
1	1	0	1	0	
1	1	1	1	1	

Table 4:Operation carrying out 8-bit adder circuit

2.4 8-bit adder parallel circuit

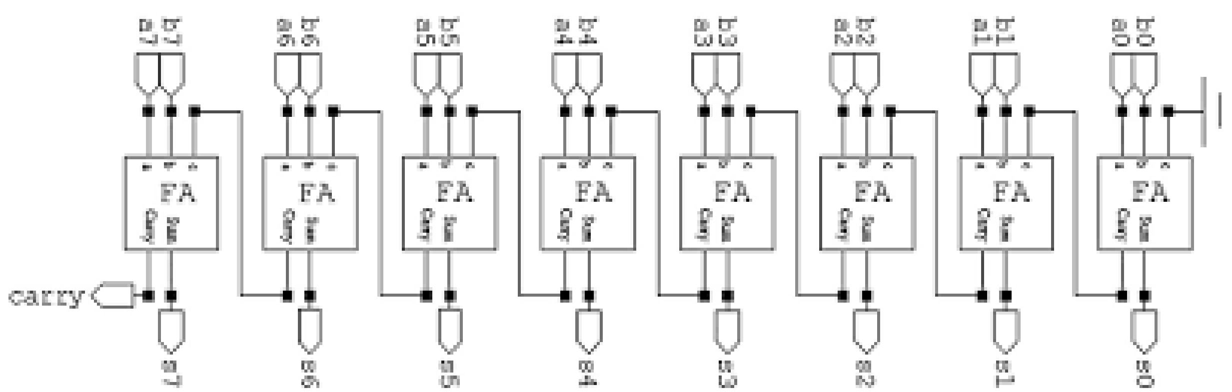

Figure 5: 8-bit adder parallel circuit

3. Algorithm

Step 1: START

Step 2: Call function BitAdder as bAdder

Step 3: print “WELCOME TO BIT ADDER APPLICATION”

Step 4: Initialize variable execute to False

Step 5: Run while loop until complete equals to False

Step 6: initialize try catch block

Step 7: taking input x with command “Enter 1st number: ”

Step 8: assign input carrier Cin=0

Step 9: if x<255 or x>0 then the message shows "Please enter a number between 0 and 255"

Step 10: assign complete as True

Step 11: "The entered value is unacceptable." If the number isn’t between 255 and 0

Step 12: assign complete as False

Step 13: Run while loop until complete equals to False

Step 14: initialize try catch block

Step 15: taking another input y with command “Enter 2nd number: ”

Step 16: if y>255 or y<0 then the message shows "Please enter a number between 0 and 255"

Step 17: set complete as true

Step 18: "The entered value is unacceptable." If the number isn’t between 255 and 0

Step 19: Call function “decimalToBinary” with input x to store it in BinaryX and input y to store it in BinaryY

Step 20: prints binary value of 1st number

Step 21: prints binary value of 2nd number

Step :22 Call a string variable value as empty string

Step 23: Call Function “byte_adder” with input BinaryX and BinaryY as string and cin & store it in value

Step : gives out put

Step 24: show message "Press Enter to Continue or Enter X to Exit: "

Step 25: take input from user

Step 26: Set exe to uppercase

Step 27: if X entered print "Thank you for using the application. Goodbye!"

Step 28: call Function BinaryAdder

Step 29: END

4. Pseudocode

4.1 Pseudocode for BitAdder:

DEFINE FUNCTION byte_adder with three parameters a, b and cin

DO

SET

output as String

FOR i **IN** range (len (b)-1,-1,-1) ; loops the code

fs = gaet.xor_Gate(int (a[i]), int(b[i]))

sum_var= gate.xor_Gate(fs, cin)

output += str(sum_var)

fco = gate.and_Gate(int(a[i]), int(b[i]))

fco2 = gate.and_Gate(fs,cin)

Co = gate.or_Gate(fco, fco2)

cin = Co

IF

cin equals to 1

DO

Output += str(cin)

END DO

END IF

Output = output[::-1]

RETURN output

END DO

4.2 Pseudocode for conversion:

```
Function binaryToDecimal
DO
 Set decimalValue AS 0
        Set initial AS 1
      FOR I in range (len (Binary)-1,-1,-1)
       DO
              DecimalValue = decimalValue + initial *int(Binary[i])
              Initial = initial*2

         return decimalValue
END DO


function decimalToBinary(n)

DO
        set temp AS list
        set Binary AS String
                 WHILE
                            n is greater than 0
       DO
              IF n%2 is not equal to 0
                     DO
                            Temp.append(1)
                     ELSE
                            Temp.append(0)
                     END DO
         END IF
              N = int(n/2)
```

FOR i in range(len(temp)-1,-1,-1)

DO

Binary += str(temp[i])

LOOP

IF len(Binary) is less than 8

DO

FOR i in range (len(Binary),8)

Binary = “0” + Binary

END DO

END IF

Return binary

END DO

4.3 Pseudocode for LogicalGates:

CALL function AND_Gate(a,b):

DO

    IF a == 1 and b == 1

        DO

            RETURN 1

        ELSE

            RETURN 0

        END DO

    END IF

END DO

Call function OR_Gate(a,b)

DO

    IF a == 0 and b == 0

        DO

            RETURN 0

        ELSE

            RETURN 1

        END DO

    END IF

END DO

Call function XOR_Gate(a,b)

DO

    IF (a == 0 and b == 1) and (a==0 and b == 0)

        DO

            RETURN 0

```
        ELSE
            RETURN 1
        END     DO
```

END IF

END DO

4.4 Pseudocode for main:

```
Import conversion as bc

Import BitAdder as BAdder


Call function BinaryAdder()
DO
 PRINT "WELCOME TO BIT ADDER APPLICATION")
          set execute as False
         WHILE execute is equal to False
                   set complete as False
                  WHILE complete is equal to False
                  DO
               TRY
                    Take input x as integer and print ("Enter 1st  Number: ")
                    Set cin as 0
 IF x is greater than 255 OR x is  less than 0
      DO
          print ("Please enter a number between 0 and 255")
  END DO
              ELSE
      DO
          set complete as True
  END DO
             END IF
                    print("The entered value is unacceptable."
 set complete as False
 WHILE complete is equal to False
                  DO
```

TRY

Take input y and print (“Enter 2nd Number: “)

IF y is greater than 255 OR y is less than 0

DO

print (“The entered Number Should be in between 0 and 255”)

END DO

ELSE

DO

set complete as True

END DO

END IF

print ("The entered value is unacceptable."

set BinaryX as BC.decimalToBinary(x)

set BinaryY as BC.decimallToBinary(y)

print (f”Binary Value of 1st number: {BinaryX}”)

print (f”Binary Value of 2nd number: {BinaryY}”)

set value as String

set Value as BAdder.byte_adder(str(BinaryX), str(BinaryY),cin)

print (f”Sum: {value}”)

show command (“press Enter to Continue or Enter X to Exit: ")

set exe as exe.upper()

IF

exe is equal to “N”

DO

set execute as True

print ("Thank you for using the application. Goodbye!")

END DO

END IF

END DO

CALL BinaryAdder()

5. Flowchart

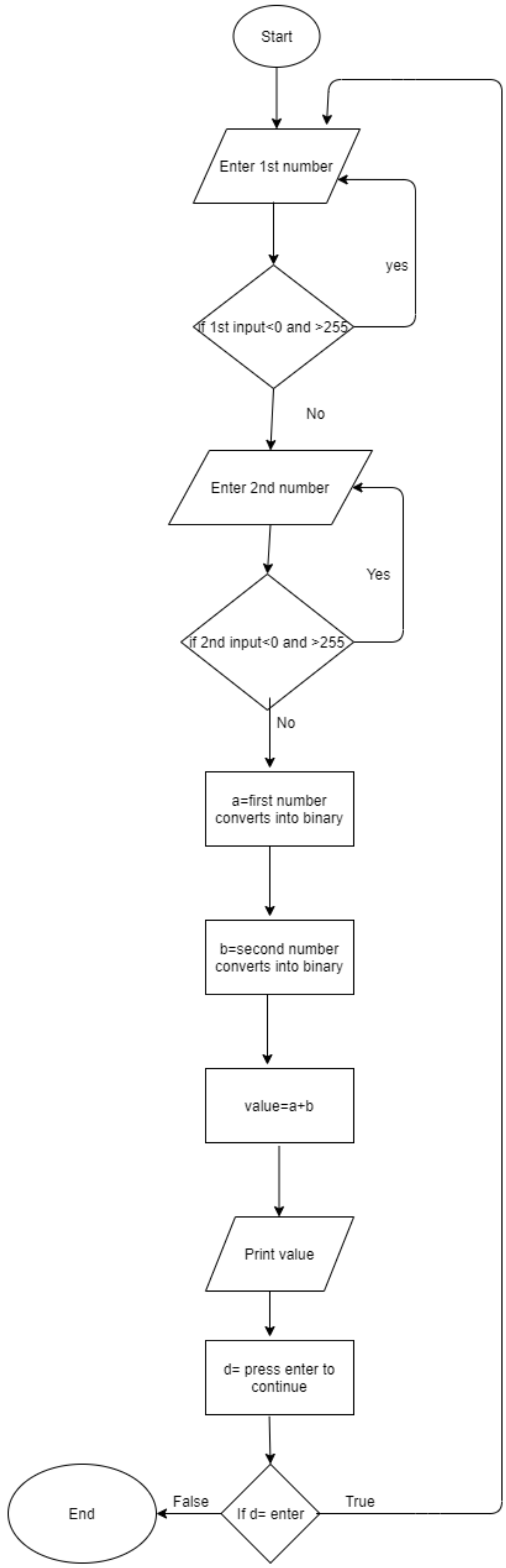

Figure 6: Flow chart

7. Data Structures

Datatypes are the classification of the type of values that the computer can take. It can also be called the classification for the data that the variable holds. The differen types of datatypes can be integer, string Boolean, float, double etc. The datatypes used in this coursework are given below:

1. String: String are the datatypes that stores alphabets, numbers or symbols. Though, numbers belong to integer but they can also be stored in string using double quotation. Same process goes to symbols too. Strings are mostly used to store keywords, names, dates etc. for instance, String var = “ Karsang”.

2. Integer: Integer is a datatype, which is set of both positive and negative number. It is one of the most commonly used datatype in every programming language. This datatype only stores numbers, be it positive or negative. It mostly used in programming languages for mathematical arithmetic operation.
   For instance, int x=10; int y=20;

3. Boolean: Boolean are the datatypes that stores only two values i.e. ‘true’ and ‘false’. This datatype is mostly used in loops in programming languages. This datatype can also be used to check different conditions.

8. Testing

8.1 Test for running the program

Objective	To run the program using command prompt.
Action	The program was attempted to open from cmd
Expected Result	The program must run from cmd.
Actual Result	The program was ran using cmd.

Table 5: Test for running program

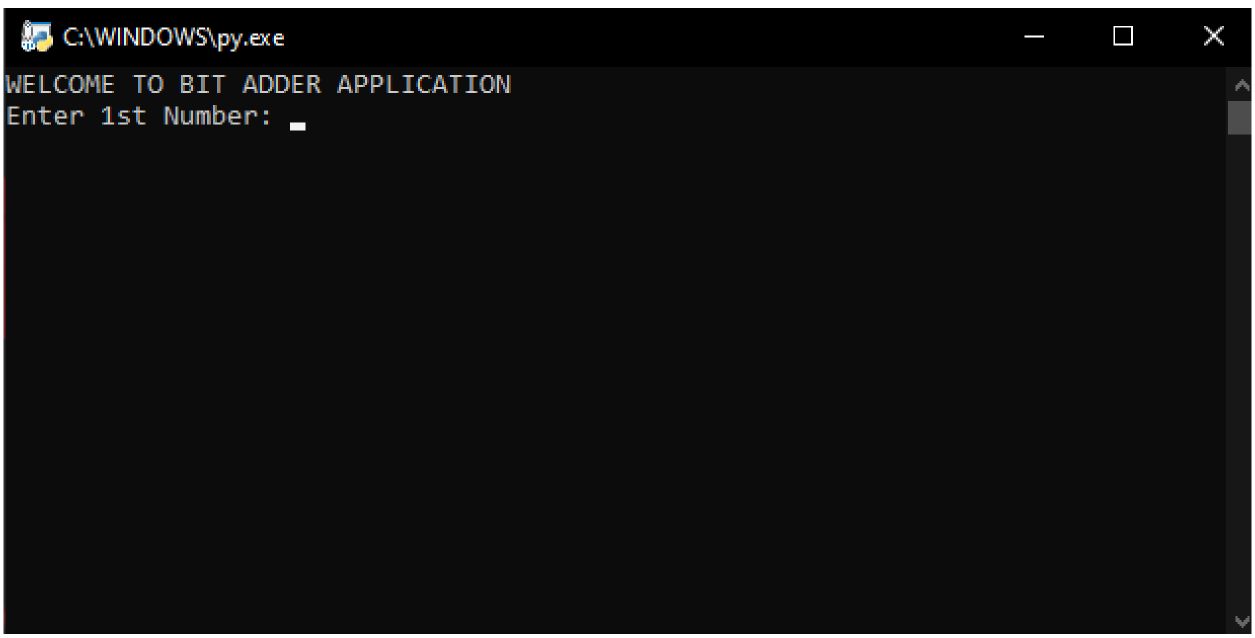

Figure 7: running program using cmd

8.2 To test if input can be more than 255 or less than 0

Objective	To test if the input can be more than 255 and less than 0
Action	Inputs more than 255 and less than 0 were entered.
Expected Result	Warning message should appear.
Actual Result	A message “Please enter a number between 255 and 0” appeared.

Table 6: Test for invalid numbers

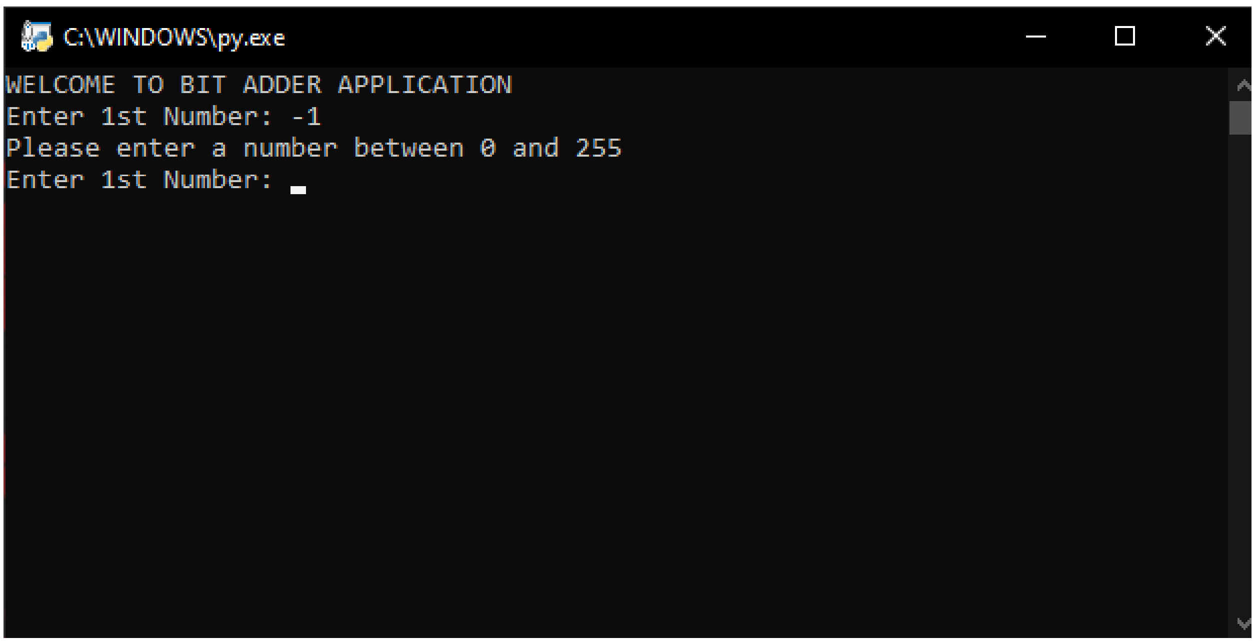

Figure 8: testing invalid numbers

8.3 To test if alphabets can be entered as input

Objective	To test if input can be alphabets.
Action	Alphabet was entered as input in the program.
Expected Result	Warning message should appear.
Actual Result	A message “the entered value is unacceptable” appeared.

Table 7: Test if alphabets can be entered as input

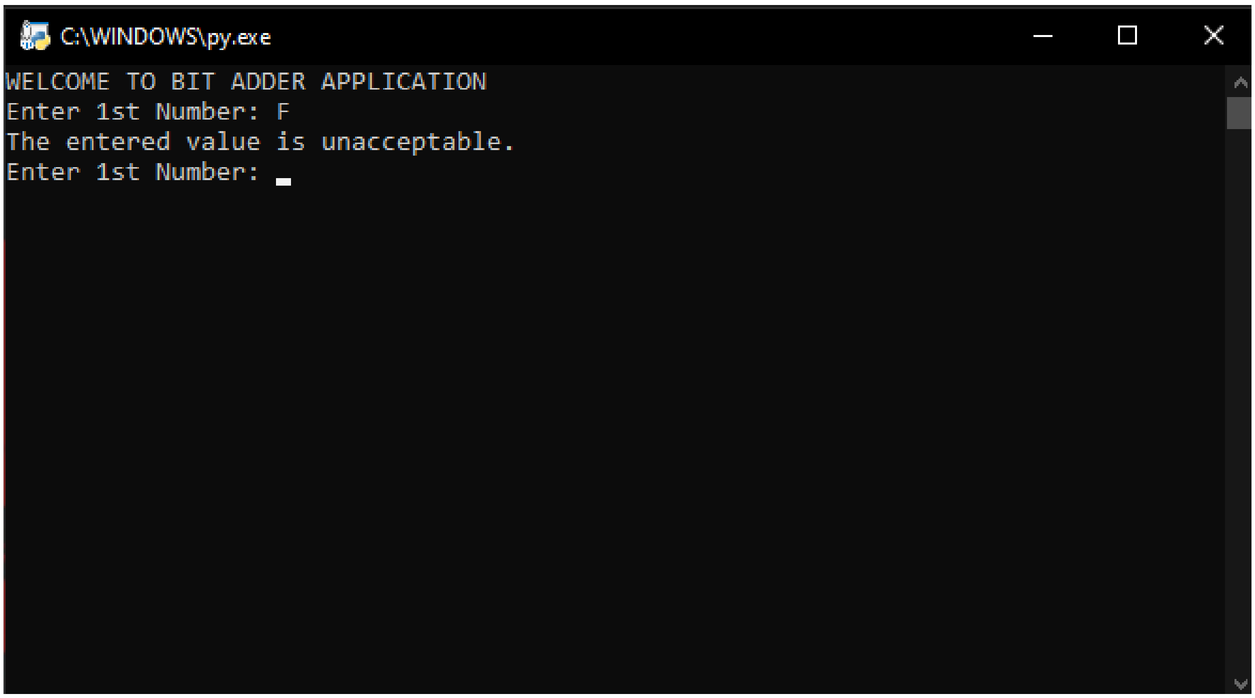

Figure 9: checking if alphabets can be entered

8.4 To test if correct output is obtained

Objective	To test if correct output is obtained
Action	Numbers 5 and 22 were entered as inputs
Expected Result	The sum of the binary numbers of the given numbers should be 00010110
Actual Result	The obtained value was 00010110.

Table 8: test if correct output is obtained

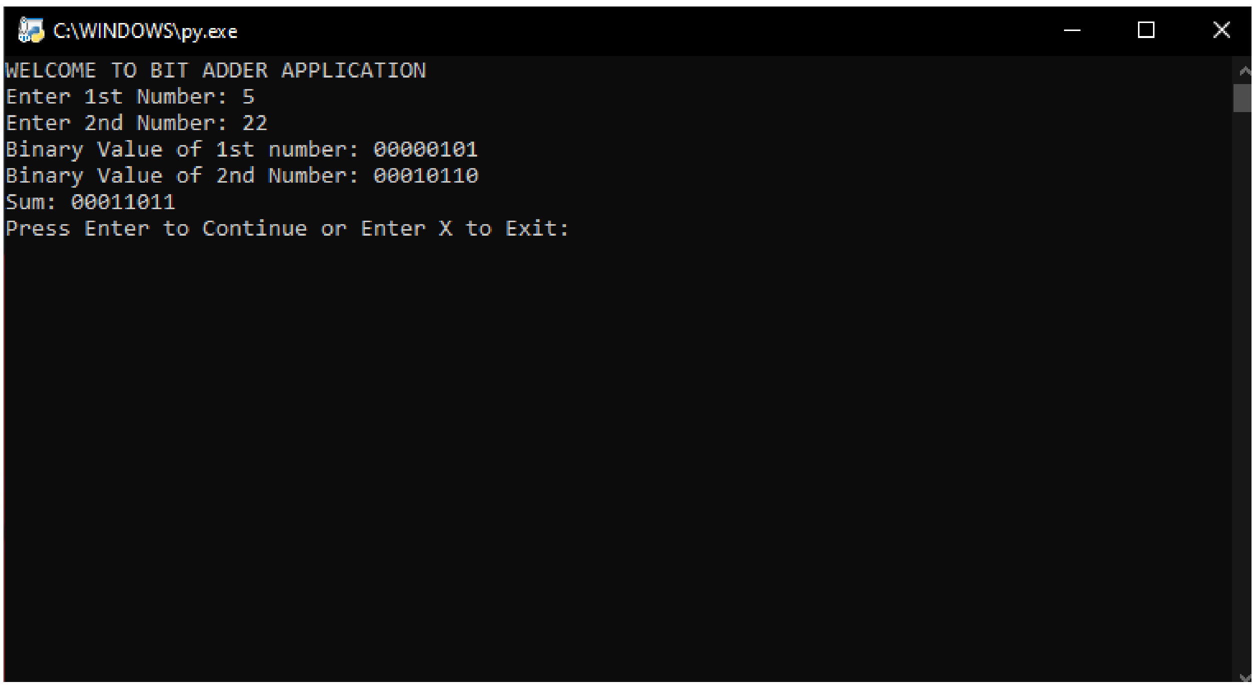

Figure 10: testing for correct output

8.5 To test if the final loop is running or not

Objective	To test if the final loop is running or not
Action	After the output was obtained, enter was pressed to run further
Expected Result	The program should again ask for input
Actual Result	Message “enter the 1st number: ” appeared.

Table 9: test if final loop is running or not

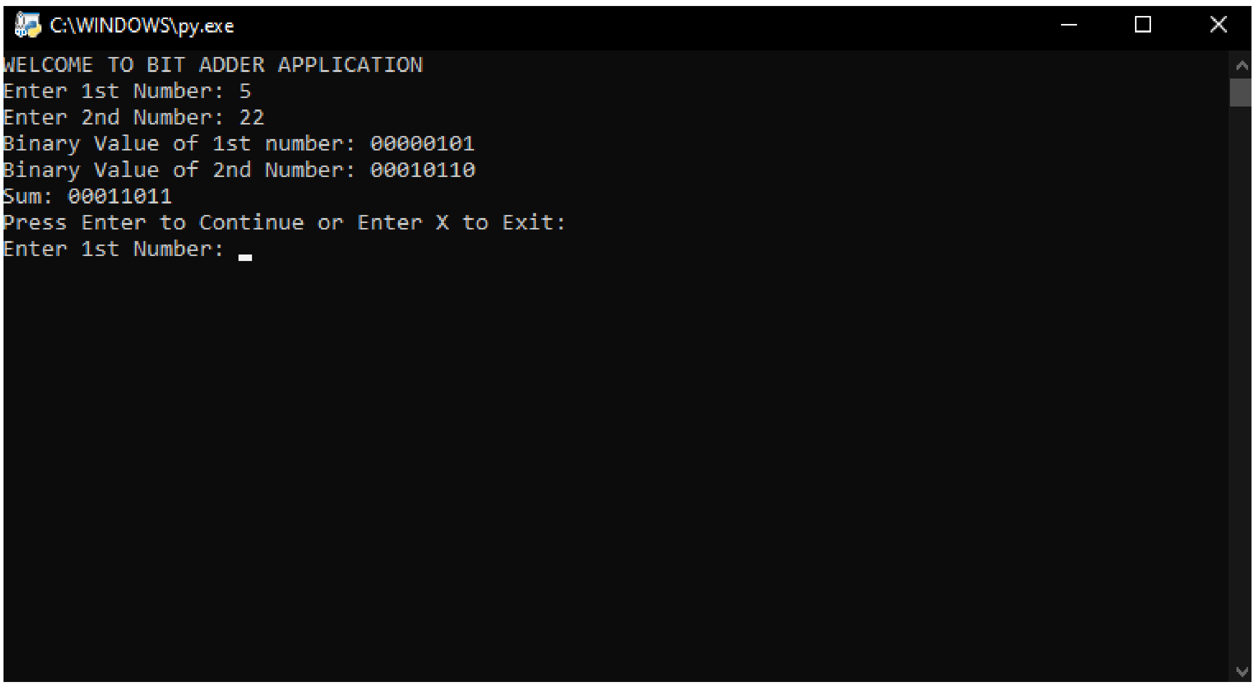

Figure 11: To check if the final loop is running or not

9. Conclusion

Firstly, I would like to heartily thank our module leaser and teachers for providing the opportunity to work on this coursework and helping throughout the module. The entire process of preparing this coursework has been really helpful to every individual of us. This coursework has helped us to gain knowledge on working with python programming language. The coursework has been able to teach us about the logic gates and the use of it. Moreover, it has taught us to work with bits and exception handling. It has also helped in working with loops and mathematical arithmetic of the bits. It has also been helpful in knowing the python programming from little pieces and working on python IDE.

10. References

Anon., 2020. [Online] Available at: https://www.python.org/doc/essays/blurb/

Anon., 2020. *Python.* [Online] Available at: https://www.python.org/doc/essays/blurb/

Subba, B. R. L., 2020. *Computer Science.* 2073 ed. s.l.:Taleju.

11. Bibliography

Anon., 2020. [Online]
Available at: https://www.python.org/doc/essays/blurb/

Anon., 2020. *Python.* [Online]
Available at: https://www.python.org/doc/essays/blurb/

Subba, B. R. L., 2020. *Computer Science.* 2073 ed. s.l.:Taleju.