LONDON metropolitan university

Numerical Data Program with Report

Andrius Uzkuraitis

ID:20028126

CS4051: Fundamentals of Computing

Contents

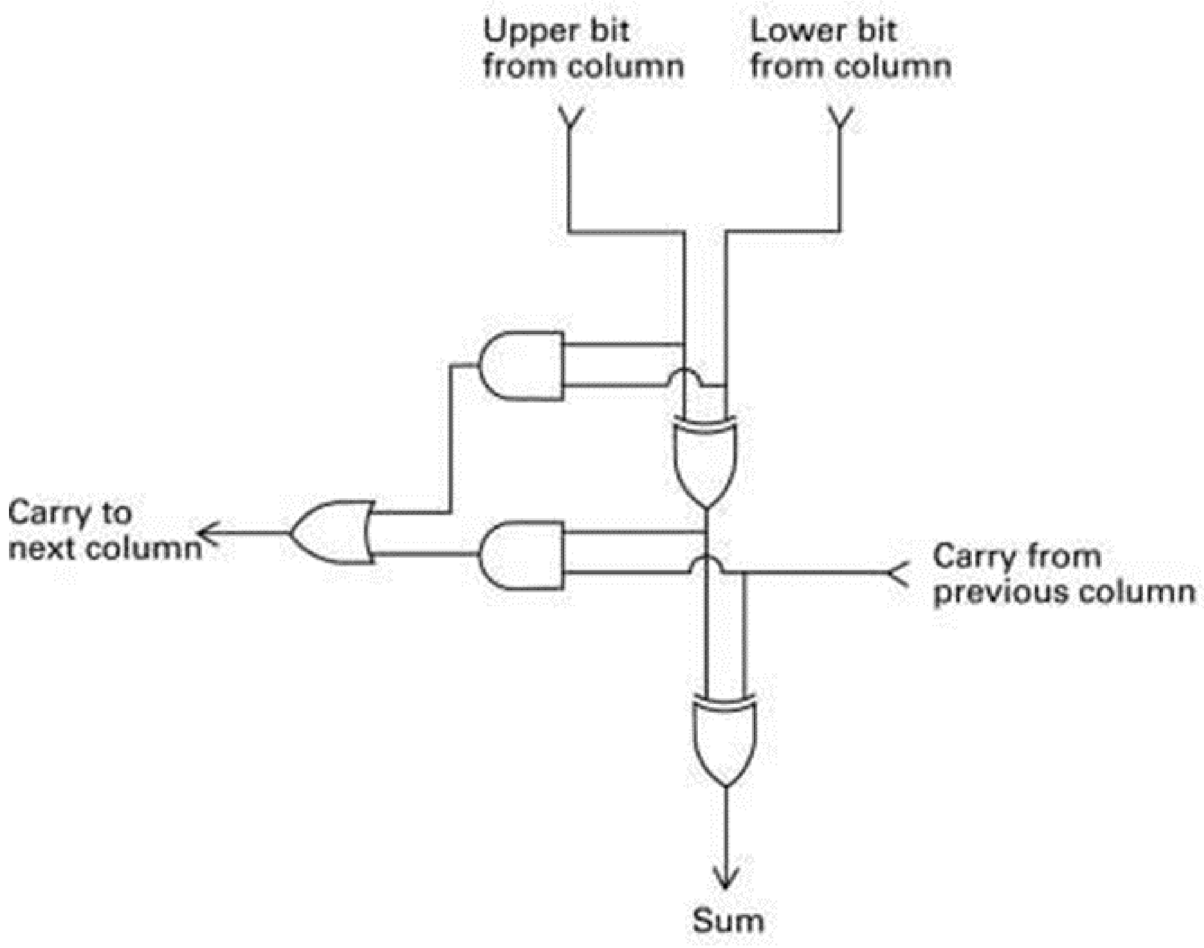
Andrius Uzkuraitis
ID:20028126

Introduction

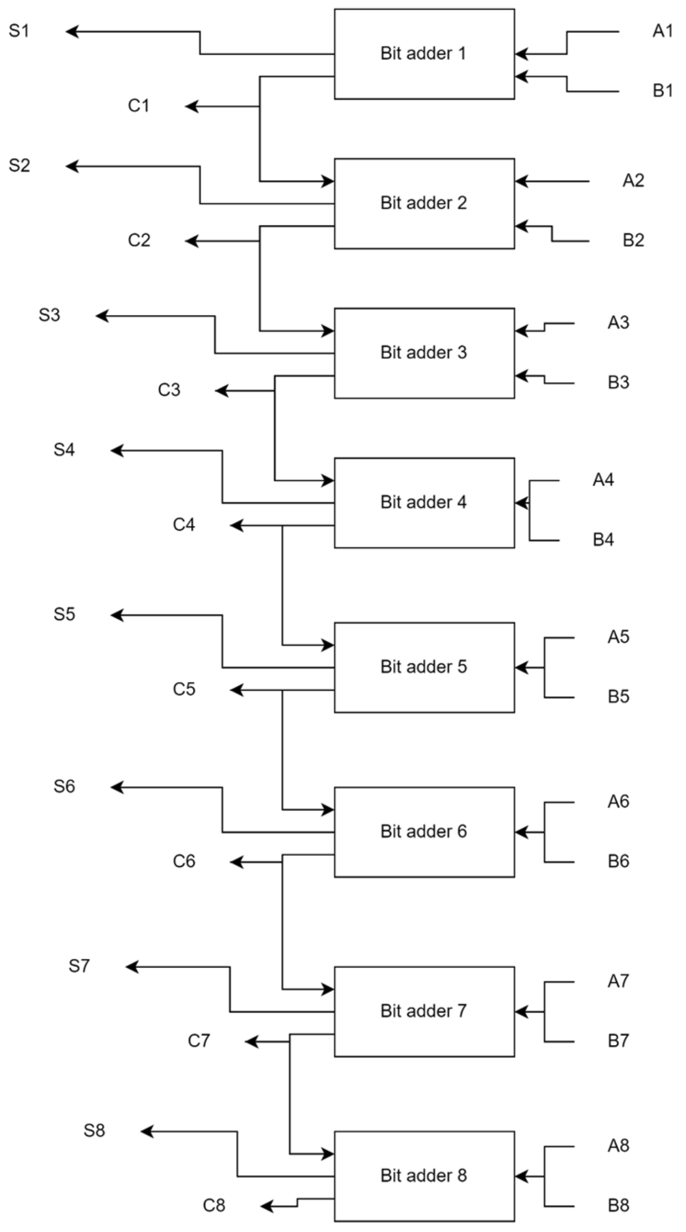
In this particular task a numerical program for bitwise addition is created that can calculate sum of two positive numbers given in binary or in decimal format and display the bitwise addition results in binary format. The program works in a loop, reads the two positive numbers and displays the sum, until it is instructed to quit by the user. The program accepts binary numbers in the range 00000000 to 11111111 and decimal number from 0 to 255 only. When any or more of the numbers are outside the specified range then program displays error message and asks the user to re-enter the numbers again. Additionally, the program also checks for other types of invalid entries like non-numeric entries and invalid format entries and displays appropriate message to user, then asks to re-enter the numbers again.

Model

Model diagram of a single bit adder:



Model of a byte adder using bit adders



In the model two-byte numbers A and B are added by using cascading combination of eight 1-bit adders. The number A is A8,…A1 where A8 is the MSB and A1 is LSB and similarly B1 is LSB and B8 is MSB of number B. The sum is C8,S8,S7,…,S1 and carries C1 to C7 are generated from their respective bit adders.

Algorithm

Program pseudocode

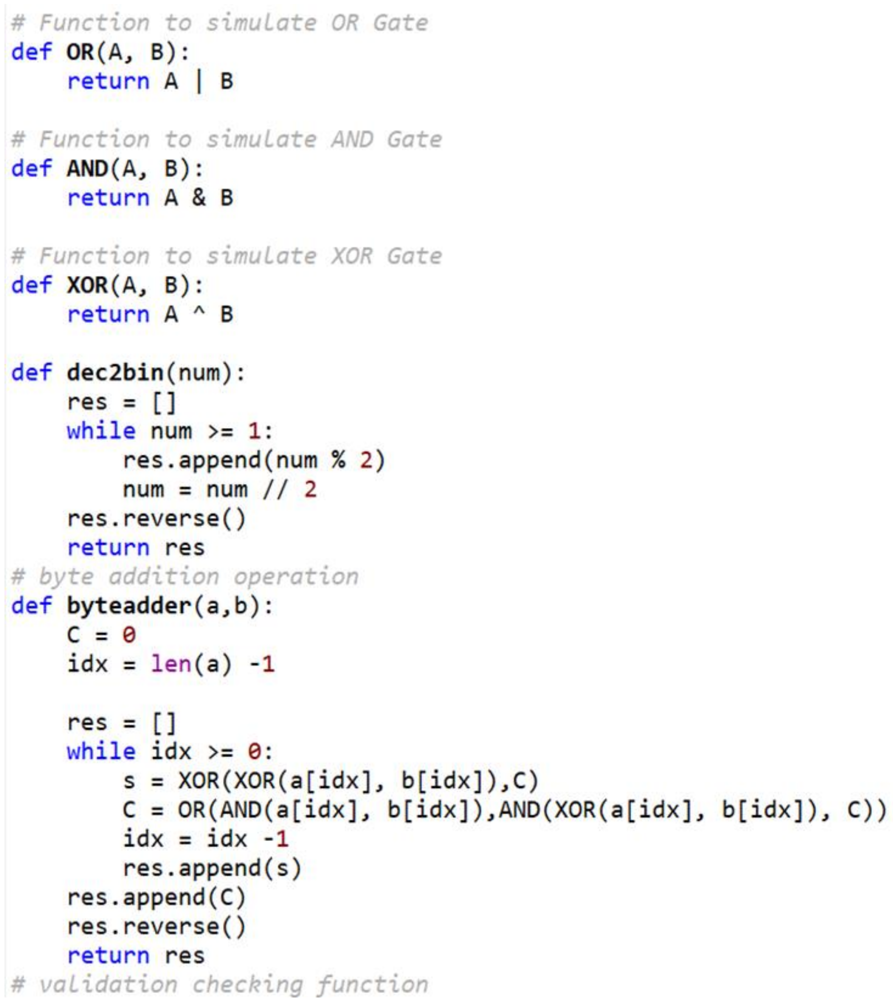
```
Define OR(A,B):
        Return A | B
Define AND(A,B):
        Return A & B
Define XOR(A, B):
  return A ^ B
Define dec2bin(num):
        Initialize res to empty list
        While num >= 1:
                Calculated remainder of num divided by 2 and append to list res
                Update num as quotient of num divided by 2
        Reverse the list res
        Return res
Define byteadder(a,b)
        Initialize C to 0
        Set idx to length of a -1
        Set res as empty list
        While idx >= 0:
                Set s as XOR(XOR(value of a at idx, value of b at idx), C)
Set C as OR(AND(value of a at idx, value of b at idx),AND(XOR(value of a at idx, value of b at idx),C))
Decrement idx by 1
Append s to res
        Append C to res
        Reverse res
        Return res
Define checkvalid(num, fmt):
        Set valid to false
        If fmt == b:
                If num is a positive integer:
                        Create set s with entries ‘0’ and ‘1’
If (length of num <= 8) and ((s == set(num)) or (set(num) == {'0'}) or (set(num) == {'1'})):
        Set valid as true; convert num to integer format from string
Else:
        Print message about invalid binary number.
                Else:
                        Print message that entered binary number is invalid
        Elseif fmt == d:
                If num is a positive integer:
                        If integer of number <= 255:
Set valid to true; convert num to binary by dec2bin() and update num
                        Else:
                                Print that decimal is over 255
                Else:
                        Print that decimal number is invalid.
        Else:
                Print that invalid format of number specified. Update valid to false.
        Return valid, num
Set user input uinp to ‘y’
While uinp == ‘y’:
        Set uinp to user input from whether to start calculation or exit program.
        If uinp == ‘y’:
                Set valinp1 false and valinp2 false
        While not (valinp1 and valinp2):
                f1 = take format input of first positive number
                inp1 = take input first positive integer
                call checkvalid(inp1,f1) and assign to valinp1 and f1
                f2 = take format of second number
                inp2 = take second positive integer input
        call checkvalid(inp2,f2) and assign to valinp2 and f2
        if valinp1 and valinp2:
            if length of num1 > length of num2:
                set nzero to length of num1 - length of num2
                append nzero number of zeros to num2
                print about appending done.
            Elif length of num1 < length of num2:
                set nzero to length of num2 - length of num1
                append nzero number of zeros to num1
                print about appending done.
            Else:
                Print that no zero-appending required.
            Call byteadder(num1,num2) and save sum in res
            Convert res from list to string
        Else:
            Print user to re-enter number as at least one of them are invalid.
    Else:
        Print that user stopped the program.
```

Data structures

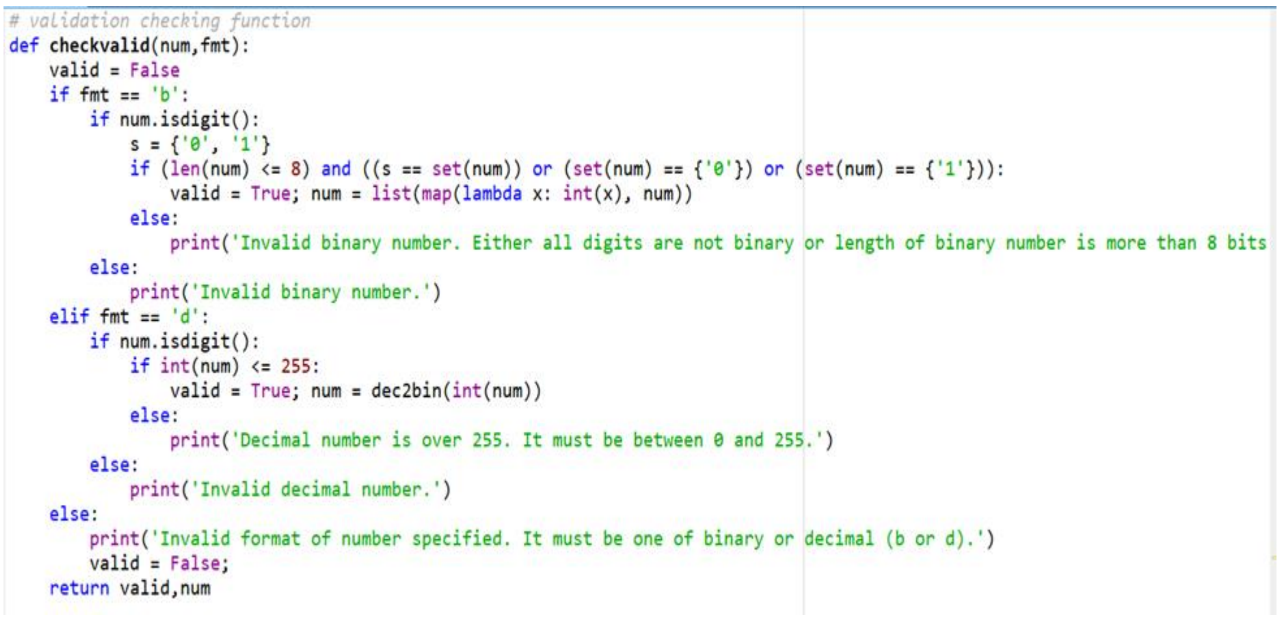
Now, while converting the above pseudocode to program the data structures like list, string, set and integer format are used. The logical operations as given in model of bit adder which are OR, AND and XOR operations are defined in the program and then called as necessary.

Program

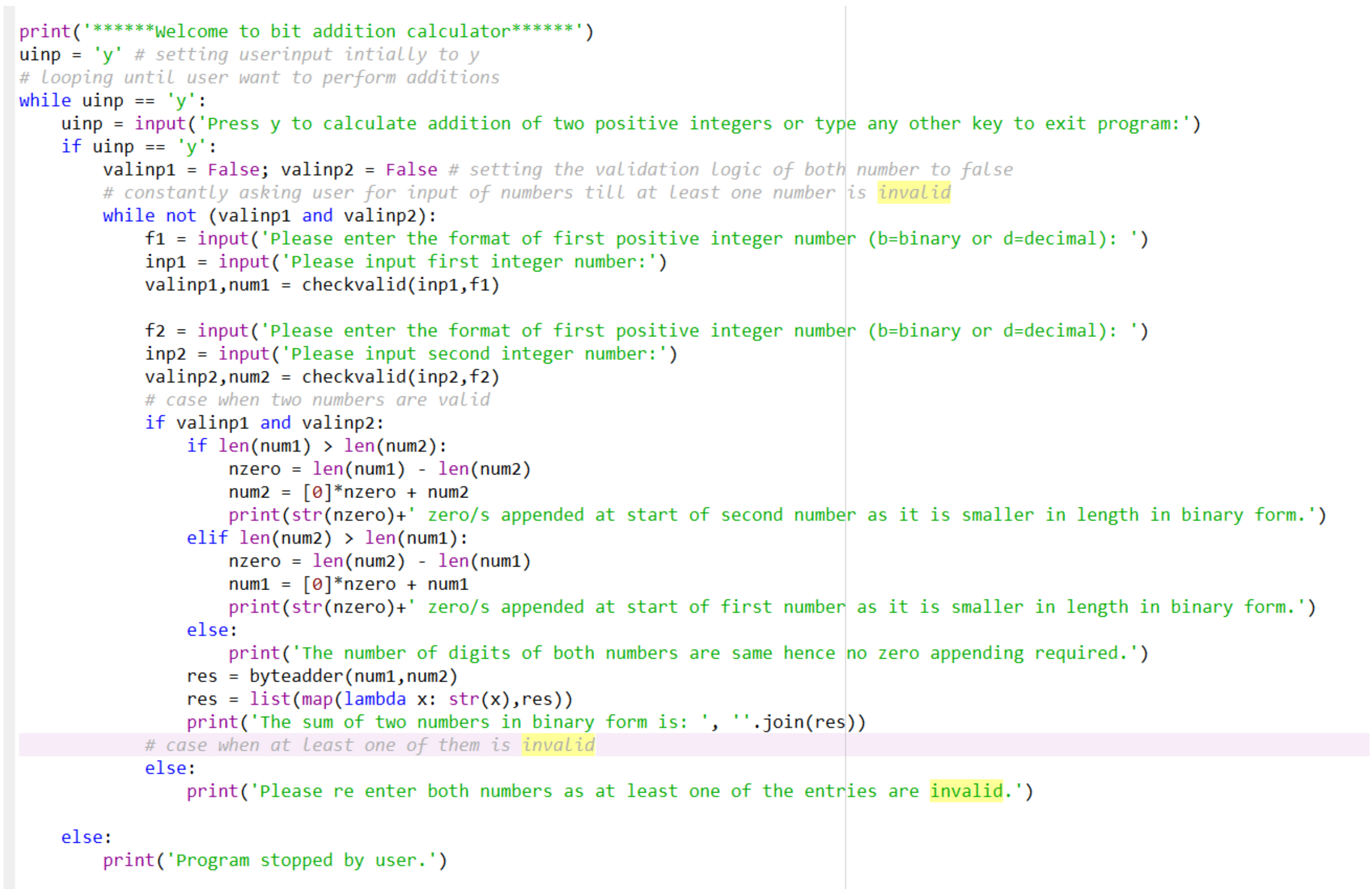
Program screenshot as developed in python.



```python
# Function to simulate OR Gate
def OR(A, B):
    return A | B

# Function to simulate AND Gate
def AND(A, B):
    return A & B

# Function to simulate XOR Gate
def XOR(A, B):
    return A ^ B

def dec2bin(num):
    res = []
    while num >= 1:
        res.append(num % 2)
        num = num // 2
    res.reverse()
    return res
# byte addition operation
def byteadder(a,b):
    C = 0
    idx = len(a) -1

    res = []
    while idx >= 0:
        s = XOR(XOR(a[idx], b[idx]),C)
        C = OR(AND(a[idx], b[idx]),AND(XOR(a[idx], b[idx]), C))
        idx = idx -1
        res.append(s)
    res.append(C)
    res.reverse()
    return res
# validation checking function
```



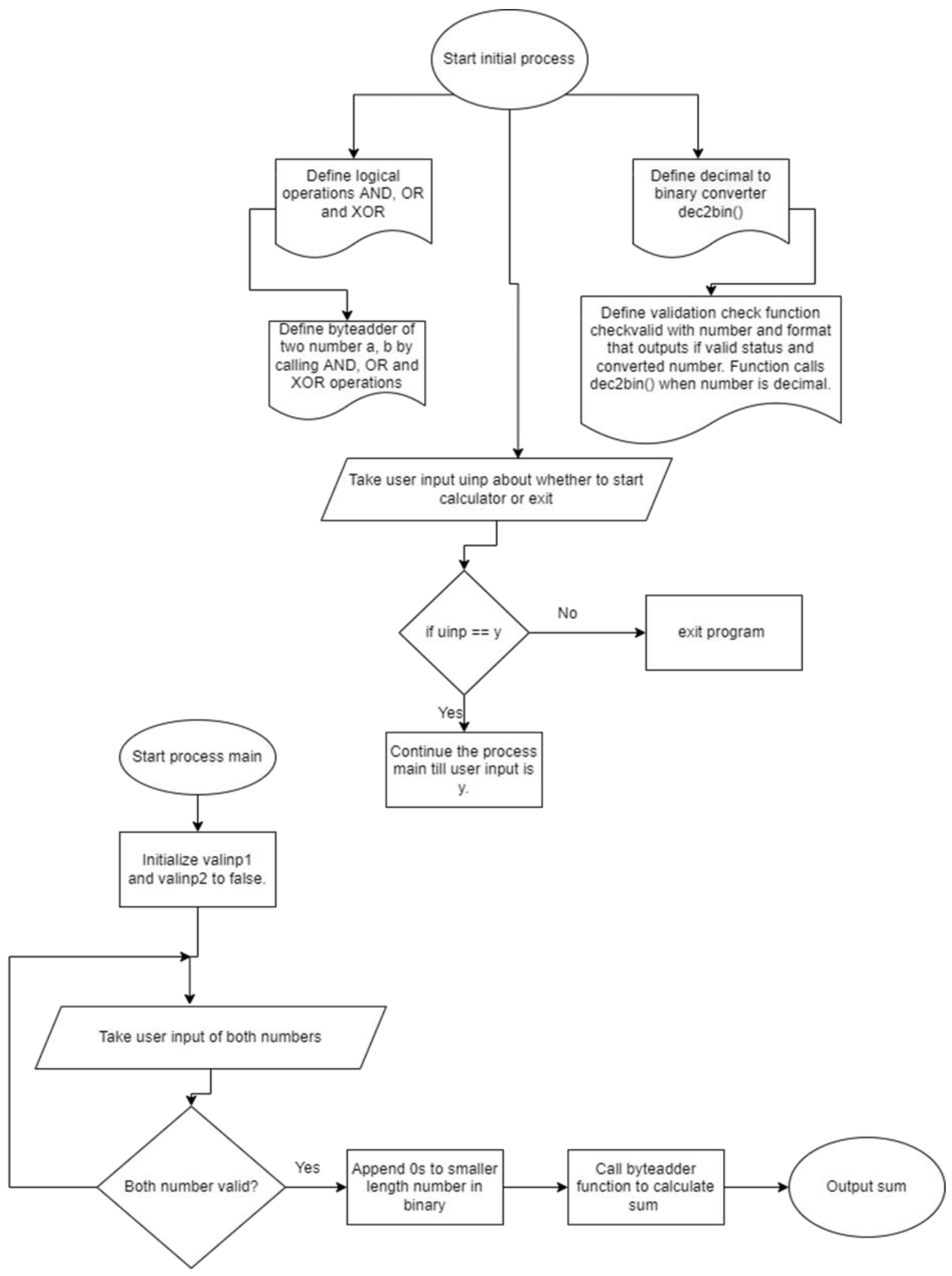
```python
# validation checking function
def checkvalid(num,fmt):
    valid = False
    if fmt == 'b':
        if num.isdigit():
            s = {'0', '1'}
            if (len(num) <= 8) and ((s == set(num)) or (set(num) == {'0'}) or (set(num) == {'1'})):
                valid = True; num = list(map(lambda x: int(x), num))
            else:
                print('Invalid binary number. Either all digits are not binary or length of binary number is more than 8 bits
        else:
            print('Invalid binary number.')
    elif fmt == 'd':
        if num.isdigit():
            if int(num) <= 255:
                valid = True; num = dec2bin(int(num))
            else:
                print('Decimal number is over 255. It must be between 0 and 255.')
        else:
            print('Invalid decimal number.')
    else:
        print('Invalid format of number specified. It must be one of binary or decimal (b or d).')
        valid = False;
    return valid,num
```



```python
print('******Welcome to bit addition calculator******')
uinp = 'y' # setting userinput intially to y
# looping until user want to perform additions
while uinp == 'y':
    uinp = input('Press y to calculate addition of two positive integers or type any other key to exit program:')
    if uinp == 'y':
        valinp1 = False; valinp2 = False # setting the validation logic of both number to false
        # constantly asking user for input of numbers till at least one number is invalid
        while not (valinp1 and valinp2):
            f1 = input('Please enter the format of first positive integer number (b=binary or d=decimal): ')
            inp1 = input('Please input first integer number:')
            valinp1,num1 = checkvalid(inp1,f1)

            f2 = input('Please enter the format of first positive integer number (b=binary or d=decimal): ')
            inp2 = input('Please input second integer number:')
            valinp2,num2 = checkvalid(inp2,f2)
            # case when two numbers are valid
            if valinp1 and valinp2:
                if len(num1) > len(num2):
                    nzero = len(num1) - len(num2)
                    num2 = [0]*nzero + num2
                    print(str(nzero)+' zero/s appended at start of second number as it is smaller in length in binary form.')
                elif len(num2) > len(num1):
                    nzero = len(num2) - len(num1)
                    num1 = [0]*nzero + num1
                    print(str(nzero)+' zero/s appended at start of first number as it is smaller in length in binary form.')
                else:
                    print('The number of digits of both numbers are same hence no zero appending required.')
                res = byteadder(num1,num2)
                res = list(map(lambda x: str(x),res))
                print('The sum of two numbers in binary form is: ', ''.join(res))
            # case when at least one of them is invalid
            else:
                print('Please re enter both numbers as at least one of the entries are invalid.')

    else:
        print('Program stopped by user.')
```

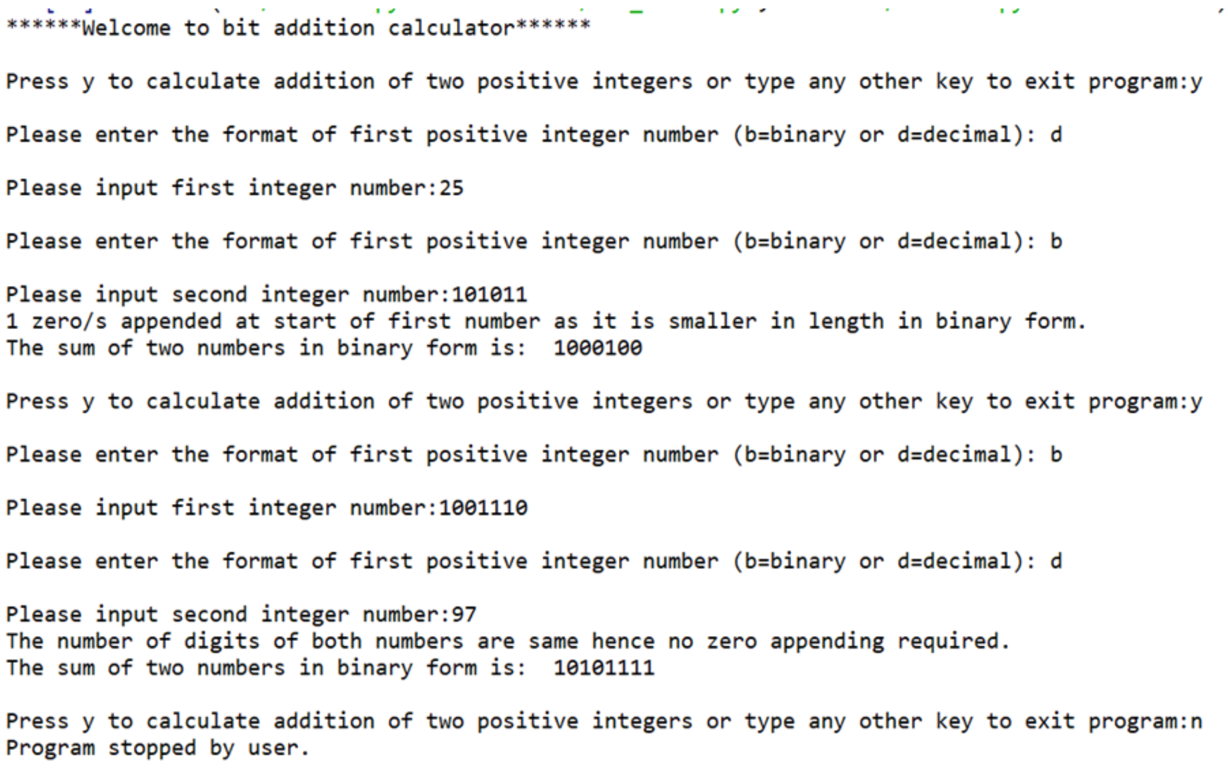
Andrius Uzkuraitis
ID:20028126

Description of program by flowchart


Start initial process
Define logical operations AND, OR and XOR
Define decimal to binary converter dec2bin()
Define byteadder of two number a, b by calling AND, OR and XOR operations
Define validation check function checkvalid with number and format that outputs if valid status and converted number. Function calls dec2bin() when number is decimal.
Take user input uinp about whether to start calculator or exit
if uinp == y
No
exit program
Yes
Continue the process main till user input is y.
Start process main
Initialize valinp1 and valinp2 to false.
Take user input of both numbers
Both number valid?
Yes
Append 0s to smaller length number in binary
Call byteadder function to calculate sum
Output sum

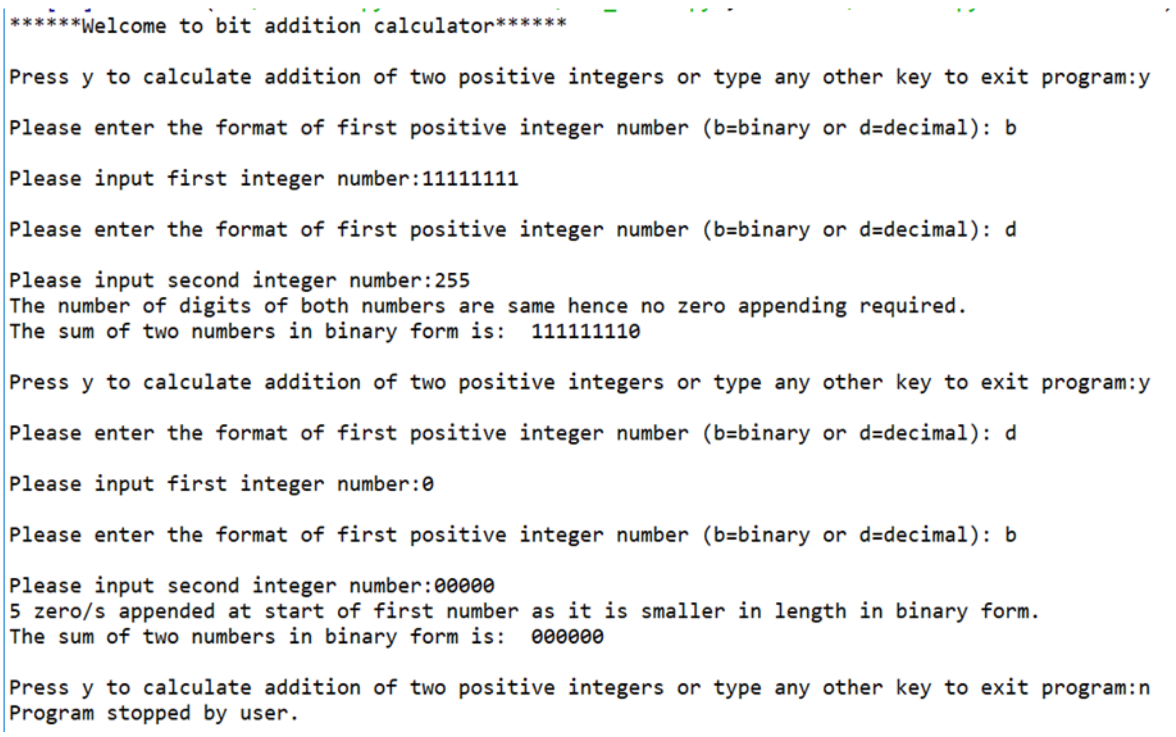
Testing

Test with normal data.



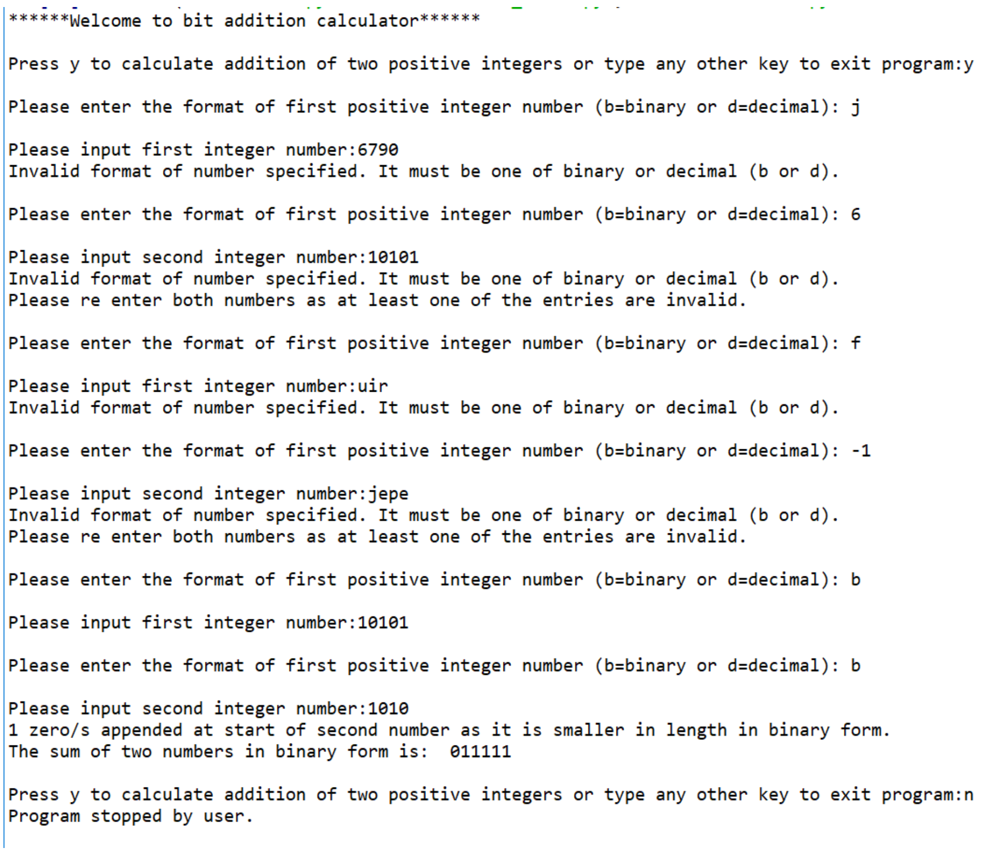
```
******Welcome to bit addition calculator******

Press y to calculate addition of two positive integers or type any other key to exit program:y

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input first integer number:25

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input second integer number:101011
1 zero/s appended at start of first number as it is smaller in length in binary form.
The sum of two numbers in binary form is:  1000100

Press y to calculate addition of two positive integers or type any other key to exit program:y

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input first integer number:1001110

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input second integer number:97
The number of digits of both numbers are same hence no zero appending required.
The sum of two numbers in binary form is:  10101111

Press y to calculate addition of two positive integers or type any other key to exit program:n
Program stopped by user.
```

Test with max/min boundary values.



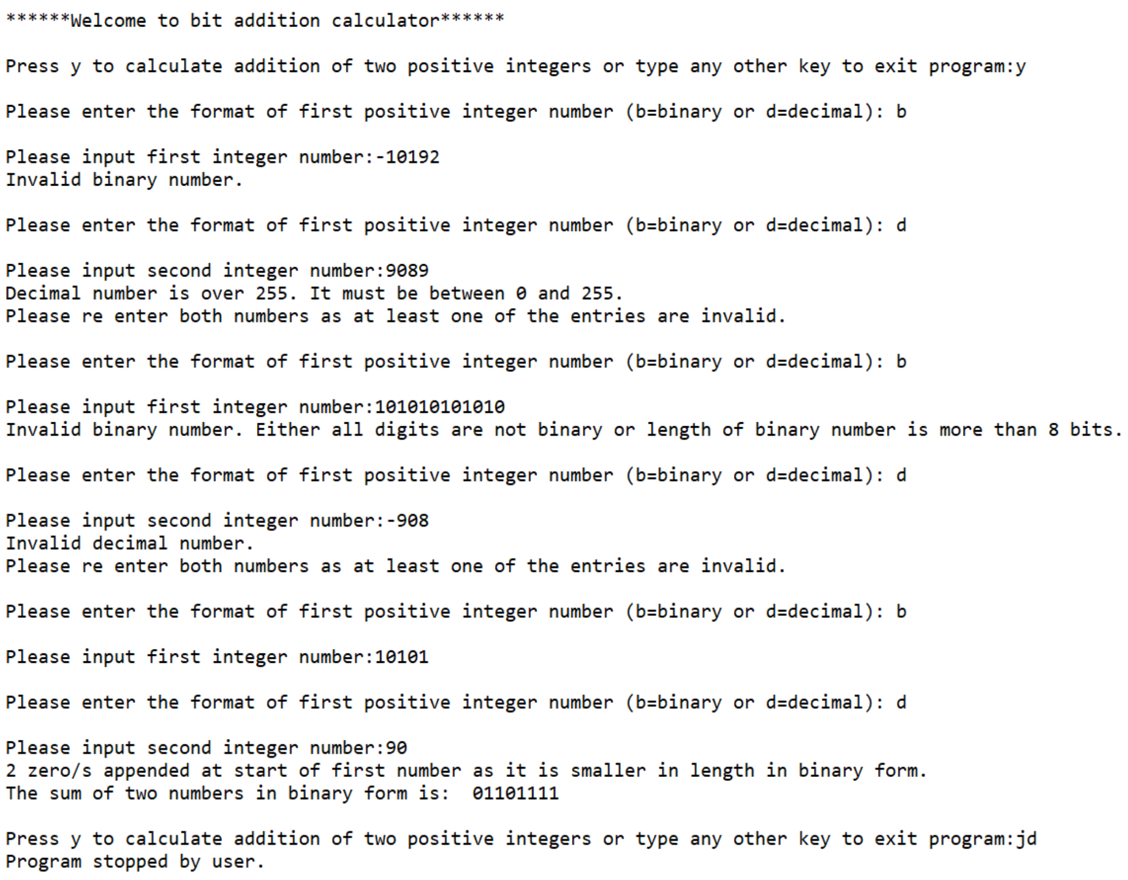
```
******Welcome to bit addition calculator******

Press y to calculate addition of two positive integers or type any other key to exit program:y

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input first integer number:11111111

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input second integer number:255
The number of digits of both numbers are same hence no zero appending required.
The sum of two numbers in binary form is:  111111110

Press y to calculate addition of two positive integers or type any other key to exit program:y

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input first integer number:0

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input second integer number:00000
5 zero/s appended at start of first number as it is smaller in length in binary form.
The sum of two numbers in binary form is:  000000

Press y to calculate addition of two positive integers or type any other key to exit program:n
Program stopped by user.
```

Test with wrong data types.



```
******Welcome to bit addition calculator******

Press y to calculate addition of two positive integers or type any other key to exit program:y

Please enter the format of first positive integer number (b=binary or d=decimal): j

Please input first integer number:6790
Invalid format of number specified. It must be one of binary or decimal (b or d).

Please enter the format of first positive integer number (b=binary or d=decimal): 6

Please input second integer number:10101
Invalid format of number specified. It must be one of binary or decimal (b or d).
Please re enter both numbers as at least one of the entries are invalid.

Please enter the format of first positive integer number (b=binary or d=decimal): f

Please input first integer number:uir
Invalid format of number specified. It must be one of binary or decimal (b or d).

Please enter the format of first positive integer number (b=binary or d=decimal): -1

Please input second integer number:jepe
Invalid format of number specified. It must be one of binary or decimal (b or d).
Please re enter both numbers as at least one of the entries are invalid.

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input first integer number:10101

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input second integer number:1010
1 zero/s appended at start of second number as it is smaller in length in binary form.
The sum of two numbers in binary form is:  011111

Press y to calculate addition of two positive integers or type any other key to exit program:n
Program stopped by user.
```

Test with wrong values.



```
******Welcome to bit addition calculator******

Press y to calculate addition of two positive integers or type any other key to exit program:y

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input first integer number:-10192
Invalid binary number.

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input second integer number:9089
Decimal number is over 255. It must be between 0 and 255.
Please re enter both numbers as at least one of the entries are invalid.

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input first integer number:101010101010
Invalid binary number. Either all digits are not binary or length of binary number is more than 8 bits.

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input second integer number:-908
Invalid decimal number.
Please re enter both numbers as at least one of the entries are invalid.

Please enter the format of first positive integer number (b=binary or d=decimal): b

Please input first integer number:10101

Please enter the format of first positive integer number (b=binary or d=decimal): d

Please input second integer number:90
2 zero/s appended at start of first number as it is smaller in length in binary form.
The sum of two numbers in binary form is:  01101111

Press y to calculate addition of two positive integers or type any other key to exit program:jd
Program stopped by user.
```

It can be observed from all the test cases as the program produces expected output as per its specification as it produces valid results in binary form when the format and entered numbers are valid and in case the numbers are invalid suitable error messages are displayed.
Comparison table of expected and obtained results when inputs are valid.

Number 1	Number 2	Obtained output	Obtained output in decimal form	Expected output in decimal form
25 (decimal)	101011(binary)	1000100	68	68
1001110(binary)	97(decimal)	10101111	175	175
11111111(binary)	255(decimal)	111111110	510	510
0(decimal)	00000(binary)	000000	0	0

Conclusion

Course work task seemed very confusing at first as I did not done any programming with Phyto besides basics. Also, the tasks given was not straight forward, so I had to re-watch the given recordings repeatedly.

The most problematic part for me was creating a function in the loop. The rest was not to hard find in the books or Google as an example of similar scenario. Although I still think that maybe I did not get it right 100% and there is still room for improvement, but the time given was too short.

References

Agrawal, A., Jaiswal, A., Roy, D., Han, B., Srinivasan, G., Ankit, A. and Roy, K., 2019. Xcel-RAM: Accelerating binary neural networks in high-throughput SRAM compute arrays. IEEE Transactions on Circuits and Systems I: Regular Papers, 66(8), pp.3064-3076.

Azeez, S. and Rangaree, P., 2021. FPGA Implementation of High Speed and Area Efficient Three Operand Binary Adder. International Journal of Advanced Science Computing and Engineering, 3(1), pp.10-17.

Jafarzadehpour, F., Molahosseini, A.S., Zarandi, A.A.E. and Sousa, L., 2019. Efficient modular adder designs based on thermometer and one-hot coding. ieee transactions on very large scale integration (vlsi) systems, 27(9), pp.2142-2155.

Muthulakshmi, S., Dash, C.S. and Prabaharan, S.R.S., 2018. Memristor augmented approximate adders and subtractors for image processing applications: An approach. AEU-International Journal of Electronics and Communications, 91, pp.91-102.

Panda, A.K., Palisetty, R. and Ray, K.C., 2020. High-speed area-efficient VLSI architecture of three-operand binary adder. IEEE Transactions on Circuits and Systems I: Regular Papers, 67(11), pp.3944-3953.