Republic of the Philippines
CAMARINES SUR POLYTECHNIC COLLEGES
Nabua, Camarines Sur

COLLEGE *of* COMPUTER STUDIES

CCCS 104 - Data Structures and Algorithms

LEARNING TASK (LINEAR DATA STRUCTURE - STACK)

GROUP NO: ___9_______		SECTION: BSCS 2A
GROUP LEADER:	Venn P. Delos Santos	
GROUP MEMBERS:	Marc Christian D.Tumaneng	
	John Mark A. Pajenago	

RATIONALE

Explain briefly Stack Linear Data Structure, how it works? What are the common examples? Its applications?
Introduce your develop Python program, what can it do?

Stacks are linear data structures that adhere to the Last In First Out rule (LIFO). This implies that the last element to be added to the stack gets eliminated first. Operations such as insertion and deletion can be performed only in one position which is the top. The most frequent applications for a stack include: Reversing a word because a stack's LIFO order results in the letters being presented in the opposite order. The value of expressions like 2 + 4 / 5 * (7 - 9) is calculated by compilers by changing the expression to prefix or postfix form. Additionally, clicking the back button in a browser stacks all the URLs you have already visited. A fresh page is added on top of the stack each time you view one. The previous URL is retrieved when you use the back button, which also removes the current URL from the stack.

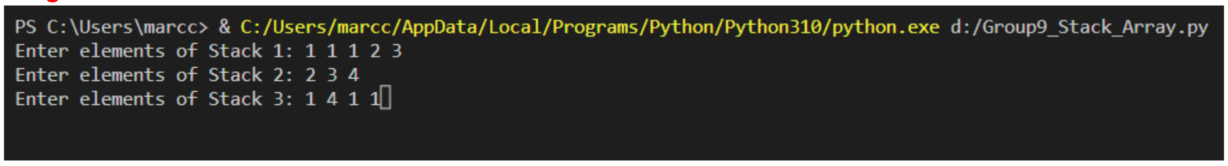
The Python program that we created uses exactly this Stack Linear Data Structure with the help and implementation of array. When the user enters the stacks' elements asked by the program, it does the push operation. It also executes the pop operation to balance the stacks and determine the height at which they will all be equal. Finally, the program prints the results and asks for confirmation from the user to restart or exit the program at the end.

USER GUIDE

Step by step instructions on how to use your program. Include images for easily visualization

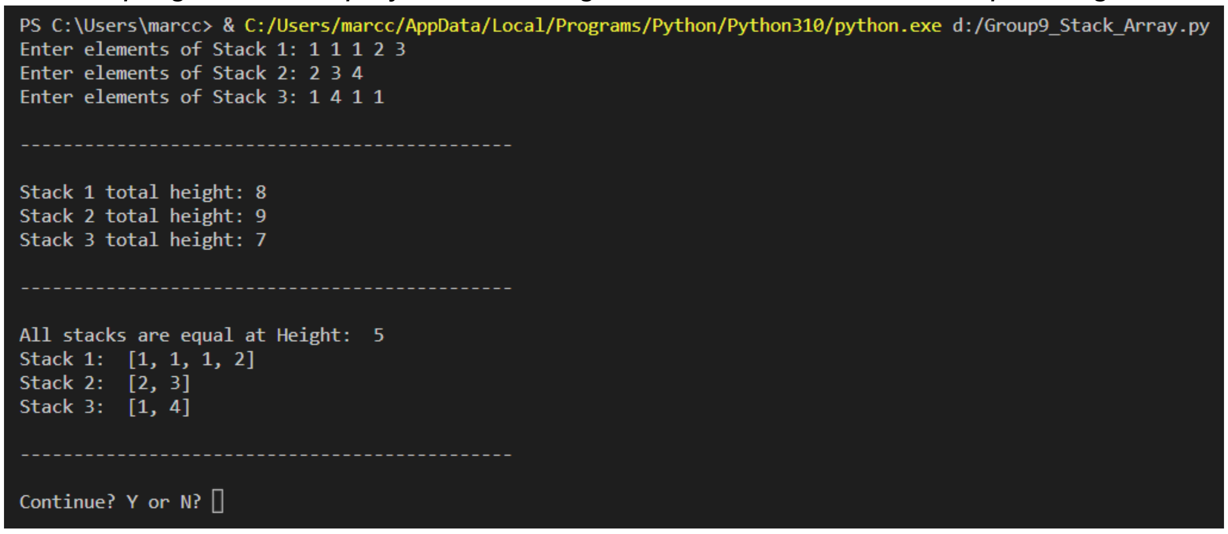
step 1
Enter the elements for Stack 1,2 ,3.

image 1



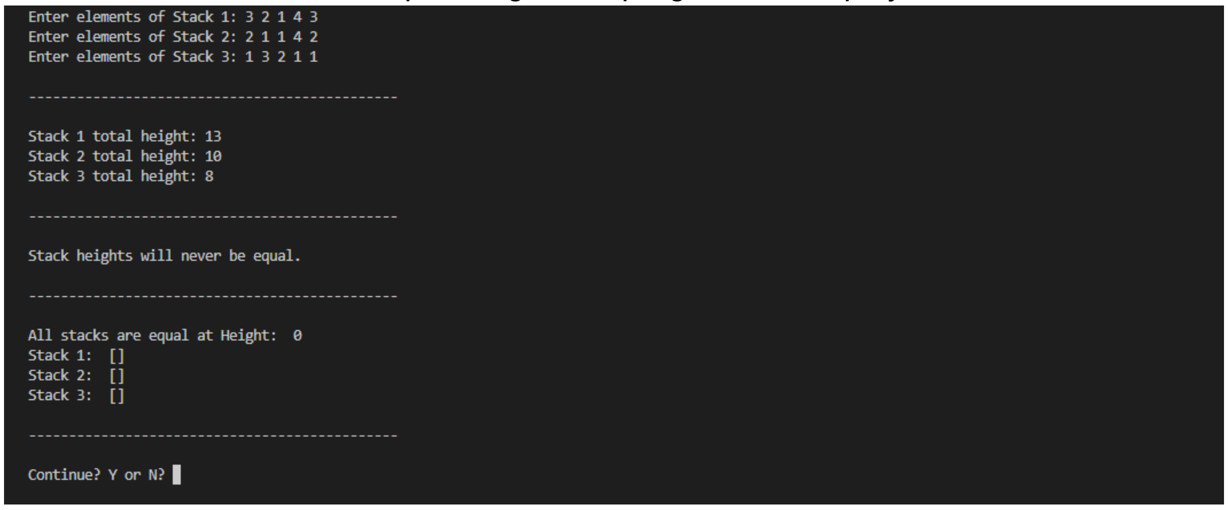
```
PS C:\Users\marcc> & C:/Users/marcc/AppData/Local/Programs/Python/Python310/python.exe d:/Group9_Stack_Array.py
Enter elements of Stack 1: 1 1 1 2 3
Enter elements of Stack 2: 2 3 4
Enter elements of Stack 3: 1 4 1 1
```

then the program will display the total height of each stack and their equal height.



```
PS C:\Users\marcc> & C:/Users/marcc/AppData/Local/Programs/Python/Python310/python.exe d:/Group9_Stack_Array.py
Enter elements of Stack 1: 1 1 1 2 3
Enter elements of Stack 2: 2 3 4
Enter elements of Stack 3: 1 4 1 1

---------------------------------------------

Stack 1 total height: 8
Stack 2 total height: 9
Stack 3 total height: 7

---------------------------------------------

All stacks are equal at Height:  5
Stack 1:  [1, 1, 1, 2]
Stack 2:  [2, 3]
Stack 3:  [1, 4]

---------------------------------------------

Continue? Y or N?
```

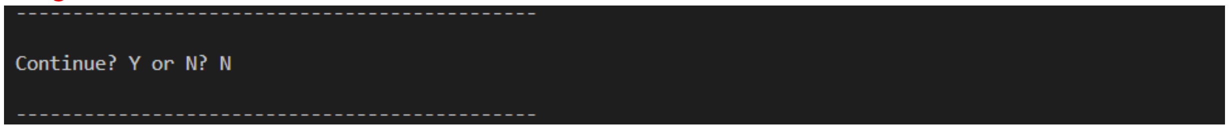
if the stacks don't have an equal height, the program will display



```
Enter elements of Stack 1: 3 2 1 4 3
Enter elements of Stack 2: 2 1 1 4 2
Enter elements of Stack 3: 1 3 2 1 1

---------------------------------------------

Stack 1 total height: 13
Stack 2 total height: 10
Stack 3 total height: 8

---------------------------------------------

Stack heights will never be equal.

---------------------------------------------

All stacks are equal at Height:  0
Stack 1:  []
Stack 2:  []
Stack 3:  []

---------------------------------------------

Continue? Y or N?
```

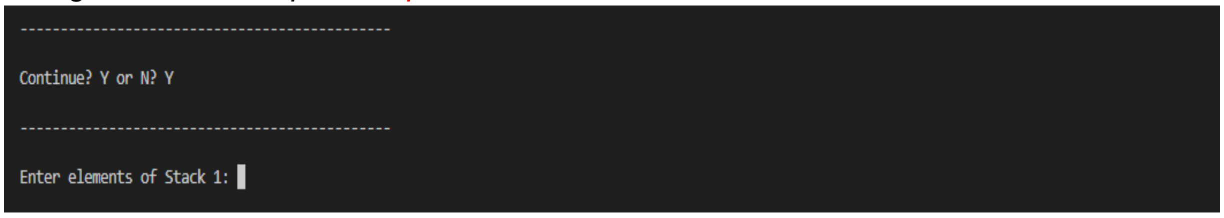
step 2
The program will ask, if continue or exit ("Continue? Y or N? ") y for yes, n for no.
**It is not case-sensitive*

image 2



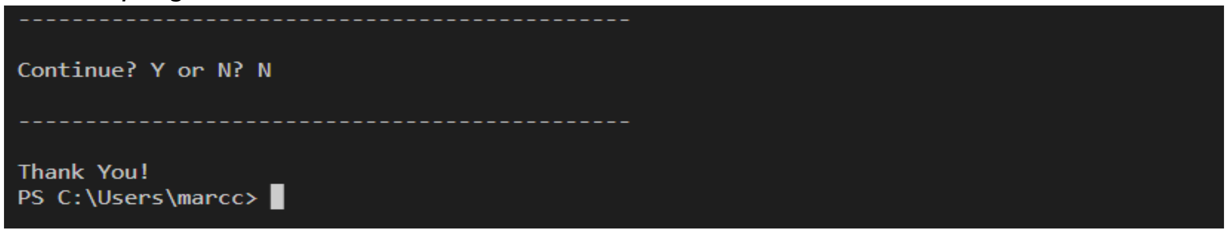
```
---------------------------------------------

Continue? Y or N? N

---------------------------------------------
```

If ‘Y’ go back to or repeat step 1



```
--------------------------------------------

Continue? Y or N? Y

--------------------------------------------

Enter elements of Stack 1: █
```

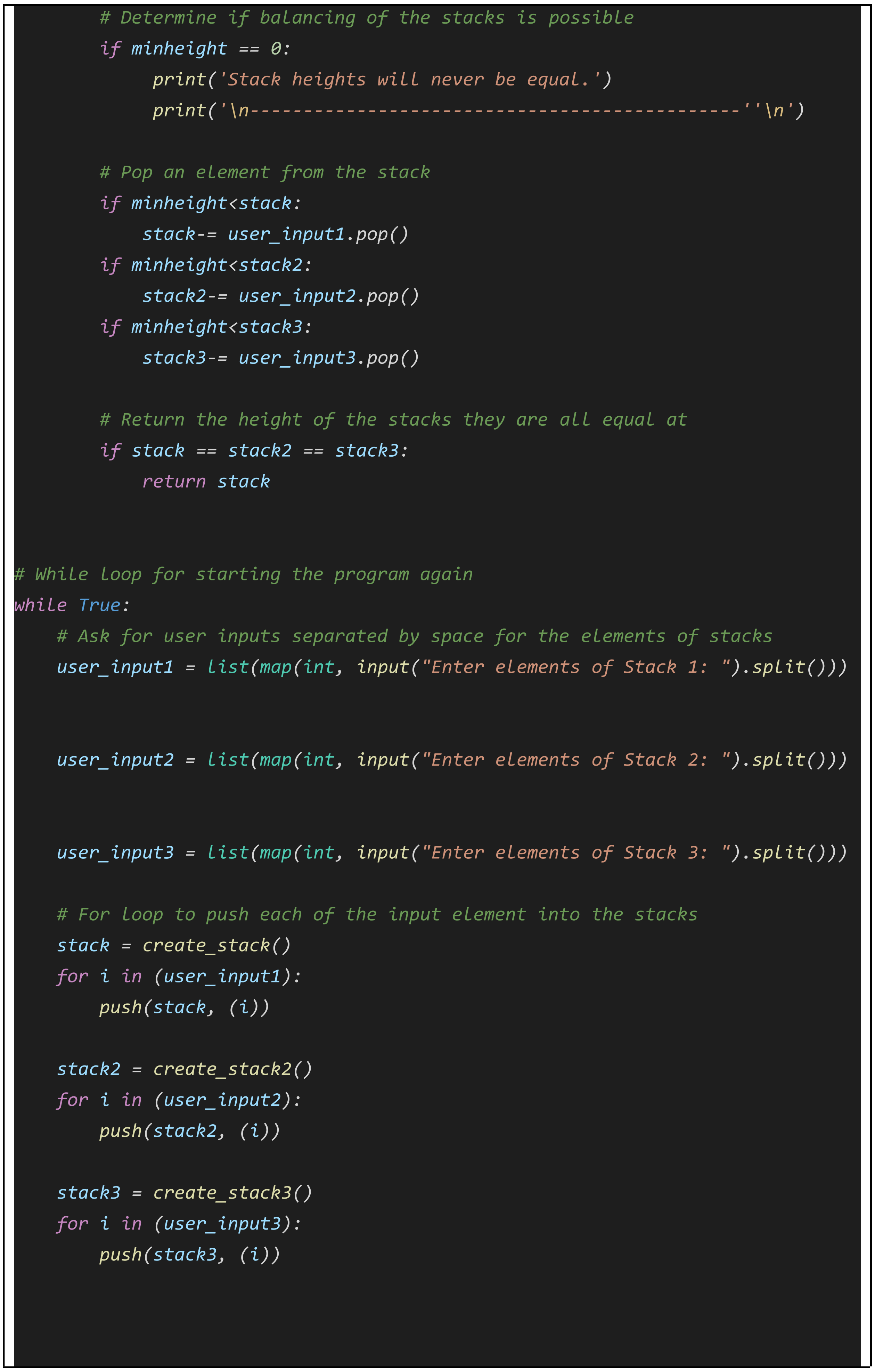
if ‘N’ the program will end.



```
---------------------------------------------

Continue? Y or N? N

---------------------------------------------

Thank You!
PS C:\Users\marcc> █
```

PROGRAM CODE

```
# BSCS 2A-GROUP 9
# Python program to demonstrate stack implementation using an array.

# Creating the array for the 1st stack
def create_stack():
    stack = []
    return stack

# Creating the array for the 2nd stack
def create_stack2():
    stack2 = []
    return stack2

# Creating the array for the 3rd stack
def create_stack3():
    stack3 = []
    return stack3

# Add items into the stacks
def push(stack, item):
    stack.append(item)

# Check the length of the stack is not empty
def check_empty(stack):
    return len(stack) == 0

# Remove an element from the stack
def pop(stack):
 if (check_empty(stack)):
    return 'stack is empty'
 return stack.pop()

# Balancing the stacks & determine which height they are all equal at
def equalStacks(user_input1, user_input2, user_input3):

    stack = sum(user_input1)
    stack2 = sum(user_input2)
    stack3 = sum(user_input3)

    while True:
        minheight = min(stack,stack2,stack3)
```



```
        # Determine if balancing of the stacks is possible
        if minheight == 0:
             print('Stack heights will never be equal.')
             print('\n----------------------------------------------''\n')

        # Pop an element from the stack
        if minheight<stack:
            stack-= user_input1.pop()
        if minheight<stack2:
            stack2-= user_input2.pop()
        if minheight<stack3:
            stack3-= user_input3.pop()

        # Return the height of the stacks they are all equal at
        if stack == stack2 == stack3:
            return stack


# While Loop for starting the program again
while True:
    # Ask for user inputs separated by space for the elements of stacks
    user_input1 = list(map(int, input("Enter elements of Stack 1: ").split()))


    user_input2 = list(map(int, input("Enter elements of Stack 2: ").split()))


    user_input3 = list(map(int, input("Enter elements of Stack 3: ").split()))

    # For loop to push each of the input element into the stacks
    stack = create_stack()
    for i in (user_input1):
        push(stack, (i))

    stack2 = create_stack2()
    for i in (user_input2):
        push(stack2, (i))

    stack3 = create_stack3()
    for i in (user_input3):
        push(stack3, (i))
    print('\n---------------------------------------------''\n')

    # Print the sums or the heights of the stacks
    print('Stack 1 total height:', sum(stack))
    print('Stack 2 total height:', sum(stack2))
    print('Stack 3 total height:', sum(stack3))


    print('\n---------------------------------------------''\n')
    # Print the stacks after balancing and the height they are all equal at
    print('All stacks are equal at Height:
',equalStacks(user_input1,user_input2,user_input3))
    print('Stack 1: ',user_input1)
    print('Stack 2: ',user_input2)
    print('Stack 3: ',user_input3)

    print('\n---------------------------------------------''\n')

    # Start again or terminate the program
    check = input("Continue? Y or N? ")
    print('\n---------------------------------------------''\n')
    if check.upper() == 'Y':
        continue
    print("Thank You!")
    break
```

TUTORIAL VIDEO

YouTube Link: https://www.youtube.com/watch?v=3HOH2D7hTEo

TAKEAWAYS

Name of Member: Venn P. Delos Santos
Contribution to the Group: coding the program, recording & documentation
Learnings: Learned how to implement Stack data structure using arrays and how to perfom operations like push and pop in a stack.

Name of Member: Marc Christian D. Tumaneng
Contribution to the Group: coding the program, recording & documentation
Learnings: I understand the concept and implementation of Stack data structures using arrays.

Name of Member: John Mark A. Pajenago
Contribution to the Group: documentation
Learnings: I learned that how to use, create, and define stack.

REFERENCES

135 - Equal Stacks | Stacks | Hackerrank Solution | Python

Implement Stack Using List | Python Tutorials | Data Structures

Programiz. (n.d). *Stack Data Structure.* https://www.programiz.com/dsa/stack#:~:text=A%20stack%20is%20a%20linear,plates%20on%20top%20of%20another.