UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 8

Stacks

Submitted by:
Ampong, J-kevin L..

Instructor:
Engr. Maria Rizette H. Sayo

October 04, 2025

I. Objectives

Introduction

A stack is a collection of objects that are inserted and removed according to the last-in, first-out (LIFO) principle.

A user may insert objects into a stack at any time, but may only access or remove the most recently inserted object that remains (at the so-called “top” of the stack)

This laboratory activity aims to implement the principles and techniques in:

- Writing Python program using Stack
- Writing a Python program that will implement Stack operations

II. Methods

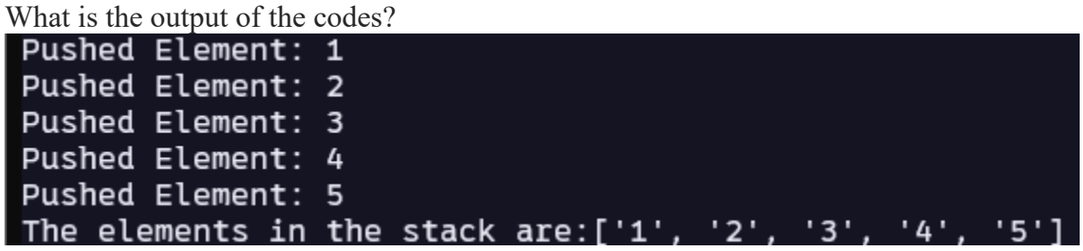
Instruction: Type the python codes below in your Colab. After running your codes, answer the questions below.

```
# Stack implementation in python


# Creating a stack
def create_stack():
    stack = []
    return stack


# Creating an empty stack
def is_empty(stack):
    return len(stack) == 0

# Adding items into the stack
def push(stack, item):
    stack.append(item)
    print("Pushed Element: " + item)

# Removing an element from the stack
def pop(stack):
    if (is_empty(stack)):
        return "The stack is empty"
    return stack.pop()

stack = create_stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
push(stack, str(5))

print("The elements in the stack are:"+ str(stack))
```

Answer the following questions:

1 Upon typing the codes, what is the name of the abstract data type? How is it implemented?
Stack, it used the list under the hood as used to hold the value append to it the implementation is simple since it did not use class instead it used direct function to create it while this good for simpler program I suggest is to class to organize different method for stack to make easy to be maintain and be updated for a near future.

2 What is the output of the codes?



3 If you want to type additional codes, what will be the statement to pop 3 elements from the top of the stack?
I will use the pop function that the code already provides, I can either write a code pop with stack as parameter value or use for loop plus range then add pop function in that loop this much clear approach than manually typing pop:

```
pop(stack)
pop(stack)
pop(stack)
# or
for _ in range(3):
    pop(stack)–
```

4 If you will revise the codes, what will be the statement to determine the length of the stack? (Note: You may add additional methods to count the no. of elements in the stack)
I would use the len() method to retrieve its total length since when you create the stack its data type is a list and data type such as list can be use the len() method to get its total size.

III. Results

```python
# Creating a stack
def create_stack():
    stack = []
    return stack


# Creating an empty stack
def is_empty(stack):
    return len(stack) == 0

# Adding items into the stack
def push(stack, item):
    stack.append(item)
    print("Pushed Element: " + item)

# Removing an element from the stack
def pop(stack):
    if (is_empty(stack)):
        return "The stack is empty"
    return stack.pop()

stack = create_stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
push(stack, str(5))

#pop three elements
for _ in range(3):
    pop(stack)

print("The elements in the stack are:"+ str(stack))
print(len(stack))
```

```
on  root  python lab8.py
Pushed Element: 1
Pushed Element: 2
Pushed Element: 3
Pushed Element: 4
Pushed Element: 5
The elements in the stack are:['1', '2']
2
```

IV. Conclusion

This lab is just for refreshing the previous topic about stack and in this lab activity the stack was created in different approaches like using pure function rather than using class to bind its method. To avoid trying to use more professional methods and organizing the method using class is more maintainable and efficient way to deal with this kind of problem.

While simpler approach is the go-to approach of many developers, we must also think about the future of the unorganized code. It will be harder to maintain once the codebase

becomes larger while simpler approach is good for prototyping, we must improve, if possible, to avoid refactoring the codebase.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.