UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 8

Stacks

Submitted by:
Catahan, Joshua A.

Instructor:
Engr. Maria Rizette H. Sayo

October 4, 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))
```

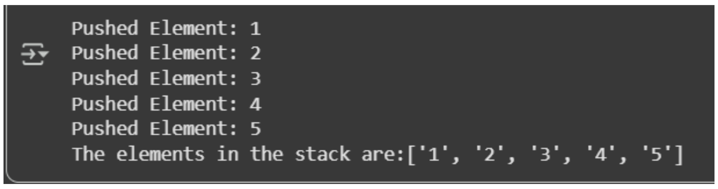
Answer the following questions:

1 Upon typing the codes, what is the name of the abstract data type? How is it implemented?
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?
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)

III. Results

1. The abstract data type used in this program is Stack. It is implemented by using the operations create_stack, is_empty, push, and pop.

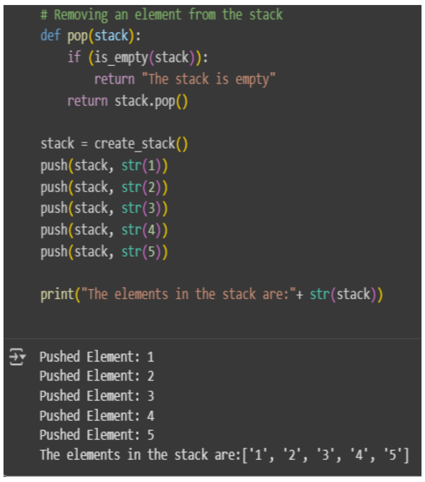
2.



```
Pushed Element: 1
Pushed Element: 2
Pushed Element: 3
Pushed Element: 4
Pushed Element: 5
The elements in the stack are:['1', '2', '3', '4', '5']
```

Figure 1: Output of the Codes

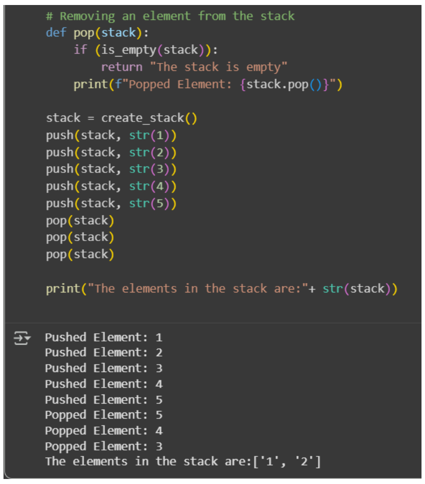
3.



```
# 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))
```

```
Pushed Element: 1
Pushed Element: 2
Pushed Element: 3
Pushed Element: 4
Pushed Element: 5
The elements in the stack are:['1', '2', '3', '4', '5']
```

Figure 2: Original Pop



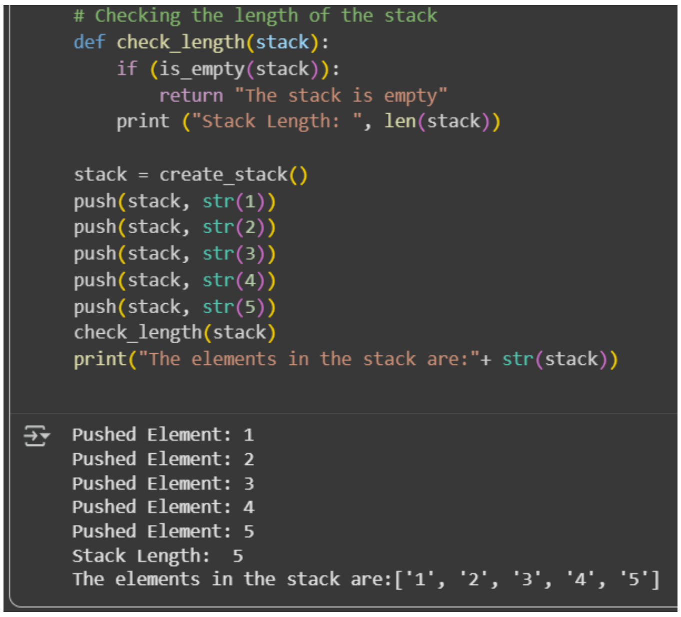
```
# Removing an element from the stack
def pop(stack):
    if (is_empty(stack)):
        return "The stack is empty"
    print(f"Popped Element: {stack.pop()}")

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

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

```
Pushed Element: 1
Pushed Element: 2
Pushed Element: 3
Pushed Element: 4
Pushed Element: 5
Popped Element: 5
Popped Element: 4
Popped Element: 3
The elements in the stack are:['1', '2']
```

Figure 3: Modified Pop

As we can see above, I modified the pop function so it will print "Popped Element: (Number)" whenever the function is called. To pop 3 items from the top of the stack I just called "pop(stack)" three times to remove the three items at the top of the stack.

4. *Figure 4: Length of the Stack*



```
# Checking the length of the stack
def check_length(stack):
    if (is_empty(stack)):
        return "The stack is empty"
    print ("Stack Length: ", len(stack))

stack = create_stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
push(stack, str(5))
check_length(stack)
print("The elements in the stack are:"+ str(stack))
```

```
Pushed Element: 1
Pushed Element: 2
Pushed Element: 3
Pushed Element: 4
Pushed Element: 5
Stack Length:  5
The elements in the stack are:['1', '2', '3', '4', '5']
```

To check the length of the stack, I created the function "check_length". It first checks if the stack is empty and then it prints the stack length after knowing that it is not empty. To call this function, I added "check_length(stack)" in the code.

Conclusion

This laboratory activity is all about Stack and it tackles the different types of operations used in this abstract data type. This was a simple and yet refreshing laboratory that has helped me understand the principles of Stack. Just like it was stated in this paper, Stack follows the last-in, first-out (LIFO) principle. It is a principle that is used in countless situations in the real world and as well as in programs, such as the undo button. In summary, I learned that the principles used in programming was designed in a way to imitate the phenomenon in the real world. This tells us that there are so many things to discover and to be created in the field of programming.

References

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