# LABORATORY 7 : Stacks

## *OBJECTIVES*

- to understand basic Stack operations
- to implement a Stack class using Python List
- to implement a simple application from the Stack class

## *BACKGROUND*

1. Stack

A LIFO (Last In First Out) data structure. A stack has one open end. It can be viewed as a pile of elements. You can operate (insert and delete) on the "top" element only. Stack in Figure 1 contains three elements. "C" is an element at the top of stack.
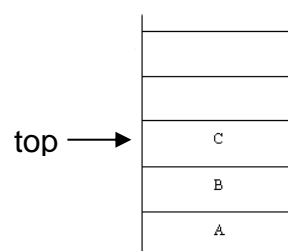


*Fig 1. Stack with three elements*

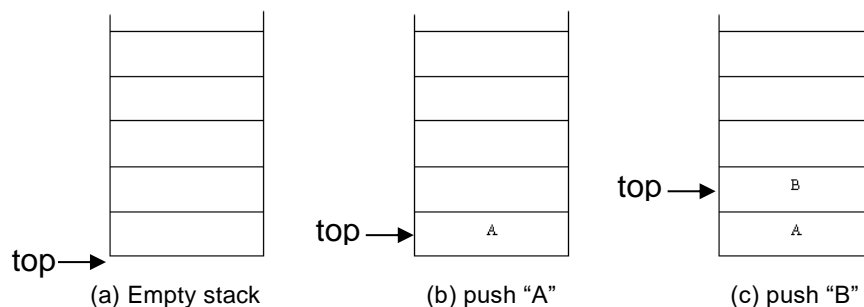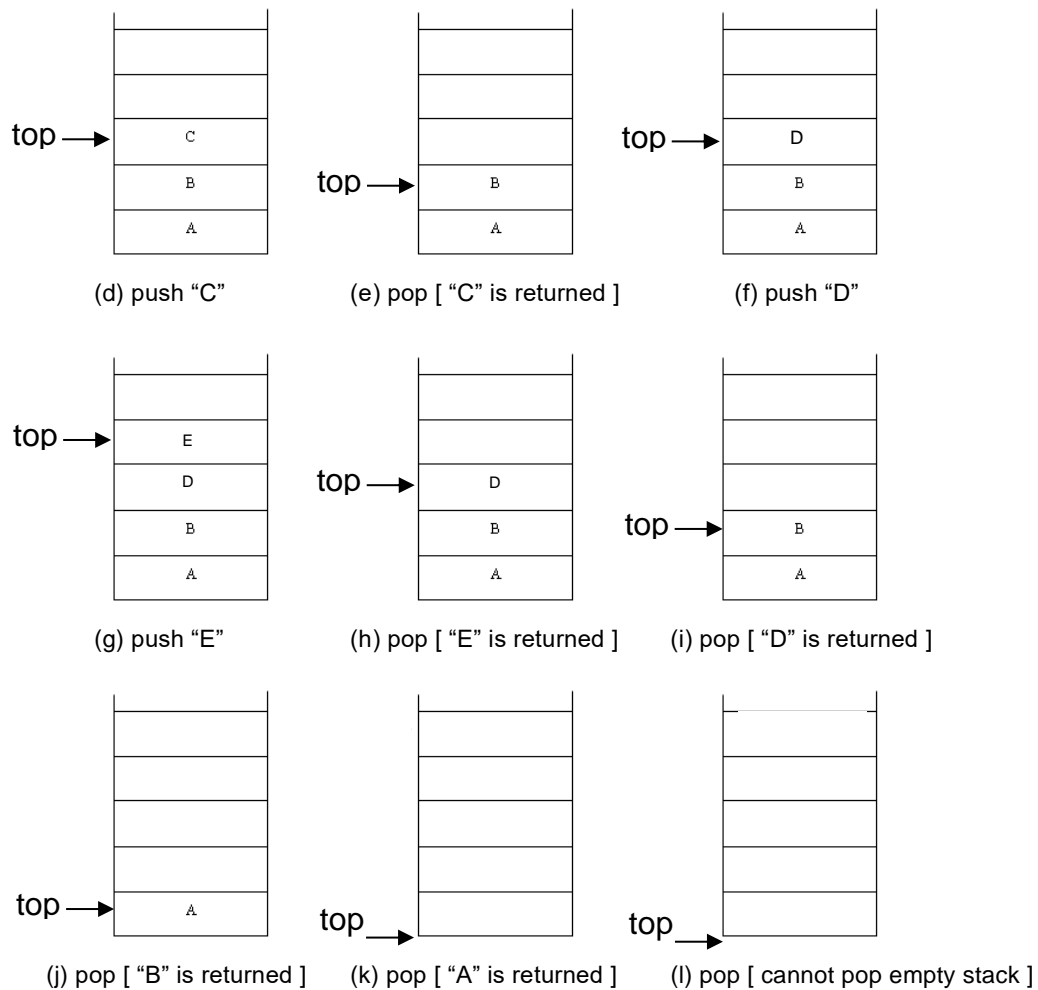A stack that contains no element is an empty stack.

2. Basic stack operations

"Push" and "pop" are two words that have special meanings when used with stacks. "Push" means insert new element into stack and "pop" means remove an element from stack.

"Push" adds one more element at stack's top. The new element will then be at the top of stack.

"Pop" remove the top element in the stack.

Figure 2 demonstrates how "push" and "pop" manipulate content of stack. Start from an empty stack in Figure 2(a) and work through series of "push" and "pop" operations. Figure 2(b)-(l) show changes in stack content after operations.



(a) Empty stack          (b) push "A"          (c) push "B"

(d) push "C"  (e) pop [ "C" is returned ]  (f) push "D"

(g) push "E"  (h) pop [ "E" is returned ]  (i) pop [ "D" is returned ]

(j) pop [ "B" is returned ]  (k) pop [ "A" is returned ]  (l) pop [ cannot pop empty stack ]

*Fig 2. Stack operations in action*

3. ADT Stack

   Data elements:
   - An ordered collection of data items that can be accessed at only one end, called the top of the stack.

   Basic operations:
   - Add new element into stack

     ```
     push(Object element)
     ```
   - Remove and return top element from stack

     ```
     pop()
     ```
   - Peek element at top of stack

     ```
     peek()
     ```
   - Check if stack is empty

     ```
     is_empty()
     ```
   - Get number of elements in stack

     ```
     size()
     ```

- Check if the element is in stack
  ```
  contain(Object element)
  ```

## *LABORATORY 7: Pre-lab*

1. Read and study Goodrich's Chapter 6 on Stacks.

## *LABORATORY 7: In-lab, Post-lab*

1. Write your own List based Stack class with all basic Stack operations

2. Valet park (as explained in class)
   - Write an algorithm (pseudocode) to simulate valet park operations.
   - Write a program to simulate valet park operations using the Stack class you wrote in 2. You may represent each car with its string ID.
   - Write a test plan and test your program according to the test plan.

Submission:

Due date: as stated in Canvas

You are to demonstrate your algorithm, test plan and program. Prepare to answer some questions individually.