LAB Manual

PART A

(PART A: TO BE REFFERED BY STUDENTS)

Experiment No.02

A.1 Aim:

To understand and implement Stacks.

A.2 Prerequisite:

Prior knowledge of introduction to data structure.

A.3 Outcome:

After successful completion of this experiment students will be able to:

After successful completion of this experiment students will be able to:

1. Understand the concept of data structures
2. Know the basics of stacks

A.4 Theory:

Data structure is logical or mathematical organization of data; it describes how to store the data and access data from memory. Actually in our programming data stored in main memory(RAM) and To develop efficient software or firmware we need to care about memory. To efficiently manage we required data structure.

There are two different types of data structure:

Linear Data Structure: In linear data structure data elements stored in sequential manner. Stack, Queue and Linked List are the types of linear data structure.

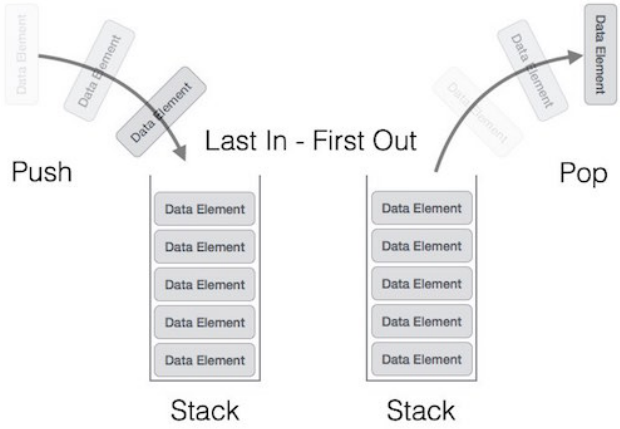
Non Linear Data Structure: In Non-Linear data structure data elements are not stored in the sequence manner. Tree and Graph are the type of non-linear data structure.

Stacks:

Stack is a linear data structure in which the insertion and deletion operations are performed at only one end.

In a stack, adding and removing of elements are performed at a single position which is known as "top".

That means, a new element is added at top of the stack and an element is removed from the top of the stack. In stack, the insertion and deletion operations are performed based on LIFO (Last In First Out) principle.



Stack is an ordered list of similar data type.

push() function is used to insert new elements into the Stack and **pop()** function is used to remove an element from the stack. Both insertion and removal are allowed at only one end of Stack called Top.

To use a stack efficiently, we need to check the status of stack as well. For the same purpose, the following functionality is added to stacks −

peek() − get the top data element of the stack, without removing it.

isFull() − check if stack is full.

isEmpty() − check if stack is empty.

At all times, we maintain a pointer to the last PUSHed data on the stack. As this pointer always represents the top of the stack, hence named top. The top pointer provides top value of the stack without actually removing it.

A.5 Procedure/Algorithm:

A.5.1 TASK 1:

Q1. Write a C/C++ program to show the implementation of Stacks using array. Show both insertion of a new element in an existing stack and deletion of an element from a stack.

Q2. Write a C/C++ program to print the reverse of a stack in the output (with or without recursion).

A.5.1 TASK 2:
Save and close the file and name it as **DSA_Exp02_YourName.**

****

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)

Roll No.	Name:
Class :	Batch :
Date of Experiment:	Date of Submission
Grade :	

B.1 Answers of Task to be written by student:

B.1.1 Code

B.1.2 Output

B.2 Observations and learning:

(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)

B.3 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.2)

****