CET 214 – Data Structures & Algorithms

Experiment # 5

Experiment Title

Stack Operations

Assessment of CLO(s): IV

Performed on 18-10-2024

Student Name			
Roll No.		**Group**	
Semester		**Session**	

Total (Max)	**Criteria 1 (2.5)**	**Criteria 2 (2.5)**	**Criteria 3 (2.5)**	**Criteria 4 (2.5)**	**Total (10)**
Marks Obtained					
Remarks (if any)					

Experiment evaluated by

Instructor's Name	**Engr. Muhammad Asad Husain**		
Date		**Signature**	

Department of Engineering Technology
(UIT University)

Course Code: CET214 Course Title: Data Structures & Algorithms Credits: 2+1 Session: Fall 2024

Rubric for assessment criteria to perform experiment number 5.

Level / Criteria	UNSATISFACTORY 1	COMPETENT 2	PROFICIENT 3	DISTINGUISHED 4
Capability of writing algorithm/ Procedure	None of the steps are implemented of an algorithm.	Few steps are implemented correctly of an algorithm.	Most of the steps are implemented correctly of an algorithm.	All the steps are implemented correctly of an algorithm.
Capability of writing Program	Programs not completed.	Completeness of code, consistent variable naming and unformatted.	Completeness of code, inconsistent variable naming and well formatted.	Completeness of code, consistent variable naming and well formatted.
Completion of target in Lab	25% target has been completed	50% target has been completed	75% target has been completed	100% target has been completed
Output	None of the outputs are correct.	Few outputs have been found correctly.	Some of the outputs are correct and well formatted.	Most of the outputs are correct and well formatted.

Practical Objective(s):

i. To implement Stacks on Linear Arrays.
ii. Insertion and Deletion in stacks.

Theory:

Stack

A stack is a list of elements in which an element can be inserted and deleted only at one end, called the top of the stack. This means, in particular, that elements are removed from a stack in the reverse order of that in which they were inserted into the stack. For this reason, stack is called a Last In First Out (LIFO) type data structure.

In order to clarify the idea of a stack let's look at a real-life example. Think of a stack of plates in a school cafeteria. When the plates are being stacked, they are added one on top of each other. It doesn't make much sense to put each plate on the bottom of the pile, as that would be far more work, and would accomplish nothing over stacking them on top of each other. Similarly, when a plate is taken, it is usually taken from the top of the stack.

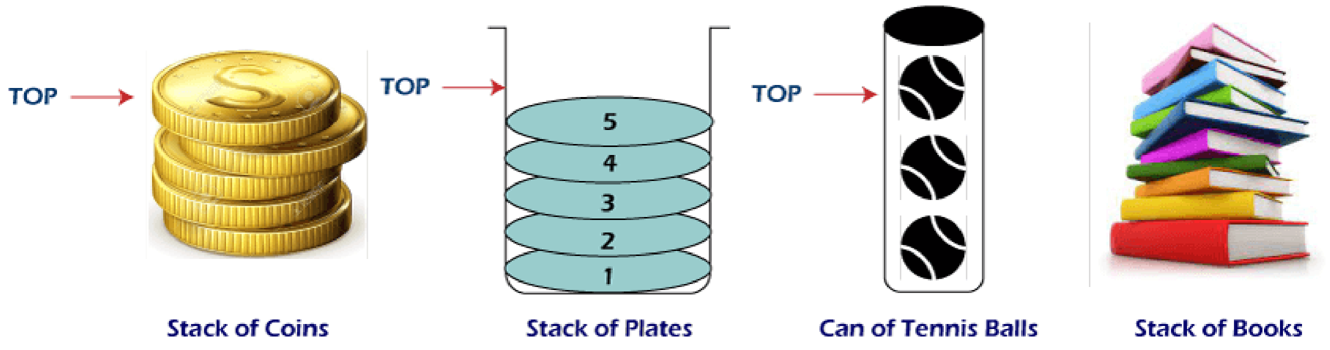
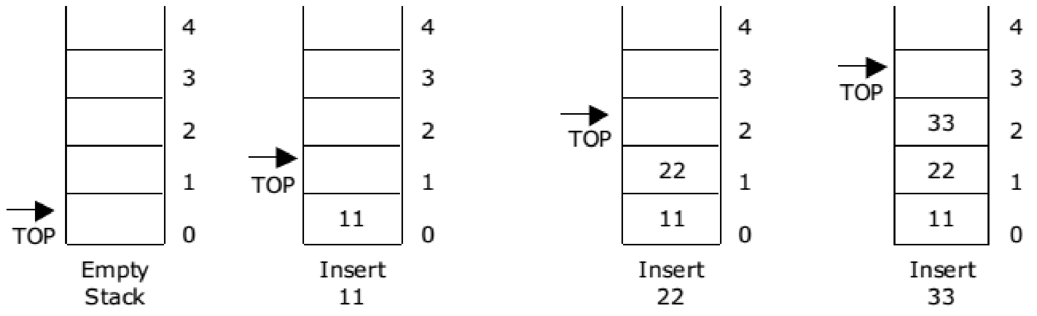


Figure 1. Real world example of Stack

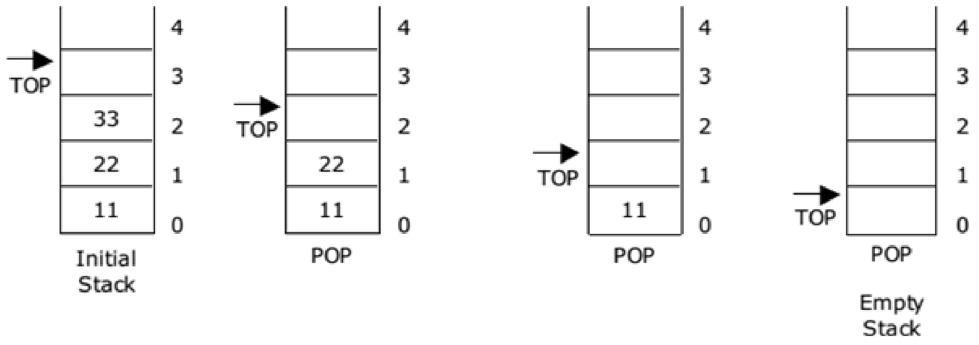
The following terminologies are used for the basic operations associated with stacks:

1. **PUSH**
   Push is the term used to insert an element into a stack. The first element pushed in goes to the bottom of the stack. The next pushed element is placed on top of first element. Similarly, each new element pushed becomes the topmost element of the stack.



2. **POP**

Pop is the term used to delete an element from a stack. It removes the topmost element of a stack. This is the only direction in which an element can be removed from a stack. If we want to delete 7 in the stack shown in Figure 2, we have to first delete 95. Only then we can delete 7.



3. **TOP**

What if you want to access the element at the top of the stack, but you do not want to remove it? It seems like a hassle (not to mention that it is inefficient) to pop the top item, use it, and push it back onto the stack. The solution to this problem is the TOP operation (do not confuse this with the variable top). Top will return the value of the top item without removing it from the stack.

Procedure 1: PUSH (STACK, TOP, MAXSTK, ITEM)

This procedure pushes an item onto a stack.

```
Step # 1: [Stack already filled?]
          If TOP=MAXSTK, then:
              Print: OVERFLOW and Return.
Step # 2: Set TOP:
          TOP = TOP+1 [Increases TOP by 1]
Step # 3: Set STACK [TOP]: = ITEM. [Inserts ITEM in new TOP position]
Step # 4: Return.
```

Procedure 2: POP (STACK, TOP)

This procedure deletes the top element of STACK and assigns 0 in its place.

```
Step # 1: [Stack has an item to be removed?]
          If TOP: = 0, then:
              Print: UNDERFLOW and Return.
Step # 2: Set STACK [TOP]: = 0. [Assigns 0 at deleted element]
Step # 3: Set TOP: = TOP – 1 [Decreases TOP by 1]
Step # 4: Return.
```

Program:

```
include<iostream>
#include<stdio.h> using
namespace std;

void PUSH(int stack1[],int top, int maxsize, int item)
{
  1. Write function definition for pop()
}
void DISPLAY(int stack1[],int top)
{
  2. Write function definition for push()
}
void DISPLAY(int stack1[],int maxsize)
{
  3. Write function definition for DISPLAY()
}

int main()
{
  int stack1[5];
  int n=5;
  int value;
  int top;

  4. Insert five different items into stack
  5. Call  DISPLAY() function
  6. Delete top two items from the stack

  DISPLAY(stack1,top);
  return 0;
  }
```

Do It Yourself:

1. Complete the program given above by implementing the given instructions.
2. Implement a function which returns the topmost element of the stack without removing it from the stack.