SVKM's NMIMS
Mukesh Patel School of Technology Management & Engineering
Computer Engineering Department
Program: B.Tech. Sem II

Course: Data Structure and Algorithm
List of Experiments
LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.02

A.1 Aim:

To study and implement concept of Stack data structure.

A.2 Prerequisite:

1. Knowledge of different operations performed on Stack data structure
2. Fundamental concepts of C\C++.

A.3 Outcome:

After successful completion of this experiment students will be able to

1. Identify the need of appropriate selection of data structure
2. Identify the steps of stack data structure selection.
3. Implement stack data structure to solve the given problem
4. Enlist the applications of stack data structure.

A.4 Theory:

A.4.1. Introduction of Stack

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last in first out) or FILO (First in last out).

The functions performed on stack are:

1. **Push:** Adds an item in the stack. If the stack is full, then it is said to be an overflow condition.

2. **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an underflow condition.
3. **Peek:** Returns the top most element from the stack.

Applications of Stack:

1. Balancing of symbols
2. Infix to postfix\prefix conversion
3. Redo-undo features at many places like editors, photoshop
4. Forward and backward feature in web browsers

A.5 Procedure/Algorithm:

A.5.1:

TASK 1:

Write a program to implement ADT of Stack.

TASK 2:

Checking for Balanced Braces in a String

Here is an example of balance braces: a{b(c)[d]e{f}}

A useful application of stacks is exactly that: to check for balanced braces in an input string.

The idea is:

You keep a Stack of braces, and every time you encounter an open brace, you push it into your stack. Every time you encounter a close brace, you pop the top element from your stack. At the end, you check your stack for being empty. If so, indeed your input string contained balanced braces. Otherwise, it didn't.

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 the there is no Black board access available)

Roll No. N230	Name: Daksh Gehlot
Class : MBA Tech CE (Sem 2)	Batch : A1
Date of Experiment: 27 January 2022	Date of Submission: 6 February 2022
Grade :	Time of Submission: 03:35 AM
Date of Grading:	

B.1 Software Code written by student: (Task 1)

(Paste your code completed during the 2 hours of practical in the lab here)

Task1:

```cpp
#include <iostream>
using namespace std;
#define max 10

// Daksh Gehlot - N230
int main()
{
    int n, num, stack[max], choice, top = -1;

    do{
        cout << "\n1. Push\n2. Pop\n3. Peek\n4. Display\n5. Exit"
        "\n\nEnter your choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            cout << "Enter the number of values: ";
            cin >> n;

            cout << "Enter the values to insert: ";
            for(int i=0;i<n;i++){
                cin >> num;
                if (top == max-1){
                    cout << "\nStack Overflow";
                } else {
                    top++;
                    stack[top] = num;
                }
            }
            break;

        case 2:
            if(top == -1){
                cout << "\nStack Underflow!";
            } else {
                cout << "\nValue deleted: " << stack[top];
                top--;
            }
            break;

        case 3:
            if(top == -1){
                cout << "\nStack is empty";
            } else {
                cout << "\nTop most value is: " << stack[top];
            }
            break;

        case 4:
            if(top == -1){
                cout << "\nStack is empty";
            } else {
                for(int i=top;i>=0;i--){
                    cout << stack[i] << endl;
                }
            }
            break;

        case 5:
            break;

        default:
            cout << "\nEnter a valid choice!";
        }
    } while (choice!=5);
    return 0;
}
```

Task2:

```cpp
#include <iostream>
using namespace std;

// Daksh Gehlot - N230
class Stack{
    int stack[10], top=-1;
public:
    int pushBrace(int val){
        top++;
        stack[top] = val;
    }
    int popBrace(){
        top--;
    }
    int isEmpty(){
        if(top <= -1)
            return true;
        else
            return false;
    }
};

int main()
{
    Stack stackItem;
    char brace[30];

    cout << "\nEnter braces to check: ";
    cin >> brace;

    for(int i=0;i<30;i++){
        if (brace[i]=='(' || brace[i]=='[' || brace[i]=='{'){
            stackItem.pushBrace(brace[i]);
        }
        else if (brace[i]==')' || brace[i]==']' || brace[i]=='}'){
            stackItem.popBrace();
        }
        else
            continue;
    }

    if(stackItem.isEmpty()){
        cout << "\nThe braces are balanced!";
    }
    else{
        cout << "\nThe braces are imbalanced!";
    }
    return 0;
}
```

B.2 Input and Output: (Task 1)

(Paste your program input and output in following format, If there is error then paste the specific error in the output part. In case of error with due permission of the faculty extension can be given to submit the error free code with output in due course of time. Students will be graded accordingly.)

Task1:

```
PS C:\NMIMS\SEM 2\DSA\Codes> cd "c:\NMIMS

1. Push
2. Pop
3. Peek
4. Display
5. Exit

Enter your choice: 1
Enter the number of values: 5
Enter the values to insert: 1 2 3 4 5

1. Push
2. Pop
3. Peek
4. Display
5. Exit

Enter your choice: 4
5
4
3
2
1
```

```
1. Push
2. Pop
3. Peek
4. Display
5. Exit

Enter your choice: 2

Value deleted: 5
1. Push
2. Pop
3. Peek
4. Display
5. Exit

Enter your choice: 3

Top most value is: 4
1. Push
2. Pop
3. Peek
4. Display
5. Exit

Enter your choice: 5
PS C:\NMIMS\SEM 2\DSA\Codes\Stack1-A2>
```

Task2:

```
PS C:\NMIMS\SEM 2\DSA\Codes\Stack1-A2> cd "c

Enter braces to check: a{b(c)[d]e{f}}

The braces are balanced!
PS C:\NMIMS\SEM 2\DSA\Codes\Stack1-A2>
```

B.3 Observations and learning [w.r.t. all tasks]:

Task 1:
We learnt:

i) Push into stack
ii) Pop out from stack
iii) Then stack was displayed using for loop and if else statements and to give user a choice to perform particular task, switch case is used.

Task 2:
Learnt how to find out if the braces in a string are balanced or not using **for loop**, **if…else statements.**

B.4 Conclusion:

Stacks are a type of container adaptors with LIFO (Last in First Out) type of working, where a new element is added at one end (top) and an element is removed from that end only. Stack uses an encapsulated object of either vector or deque (by default) or list (sequential container class) as its underlying container, providing a specific set of member functions to access its elements. Stack Syntax: - For creating a stack, we must include the header file in our code. We then use this syntax to define the std:stack:

B.5 Question of Curiosity

(To be answered by student based on the practical performed and learning/observations)

Checking for palindrome string

step 1. start
step 2. read the string from the user
step 3. calculate the length of the string
step 4. set rev = " " [empty string]
step 5. set i = length – 1
step 6. repeat until i>=0:
6.1: rev = rev + character at position 'i' of the string
6.2: i = i – 1
step 7. if string = rev
write "given string is palindrome"
step 8. else
write "given string is not palindrome"
step 9. stop

****