**DATA STRUCTURES**

# SPRING 2023

**LAB 09**

## Learning Outcomes

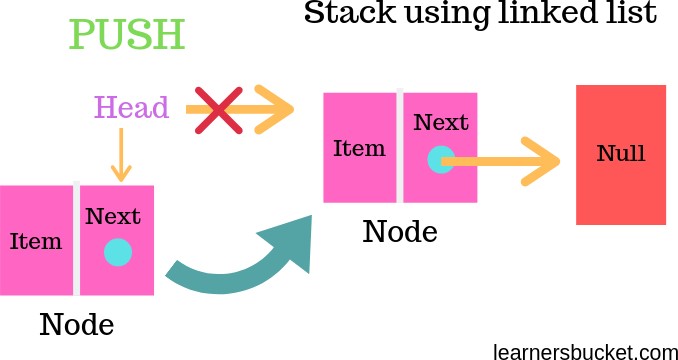
In this lab you are expected to learn the following:

**• Implement the Stack using linked list and arrays** • Google testing

**Structure**

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).

Implement a [stack u](http://www.geeksforgeeks.org/stack-data-structure/)sing single linked list in which all the single [linked list o](http://www.geeksforgeeks.org/data-structures/linked-list/)perations perform based on Stack operations LIFO (last in first out) and with the help of that knowledge we are going to implement a stack using single linked list. We are storing the information in the form of nodes, and we need to follow the stack rules and we need to implement using single linked list nodes so what are the rules we need to follow in the implementation of a stack a simple rule that is last in first out. All the operations we should perform with the help of a **top** variable.



### Lab Tasks

This section will provide more practice exercises which you need to finish during the lab. You need to finish the tasks in the required time.

**Note:** Implementation of all the variables should be template-based.

**Lab Task 1 [ Link List Based Task]**

#### Implement Following Operations

##### 1) Stack (int ignored = 0)

*Requirements:* None

*Results:* Constructor. Creates an empty stack.

##### 2) ~Stack ()

*Requirements:* None

*Results:* Destructor. Deallocates (frees) the memory used to store a stack.

##### 3) void push (const DataItem)

*Requirements:* None

*Results:* Push the element at top of the stack.

##### 4) Void pop ()

*Requirements:* Stack is not empty *Results:* Returns the element from the top of the stack.

#### 5) element Peek ()

*return element at the top of stack*

##### 6) void clear ()

*Requirements:* None

*Results:* Removes all the elements from a stack.

##### 7) Bool isEmpty ()

*Requirements:* None

*Results:* Returns **true** if a stack is empty. Otherwise, returns **false**.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Test Plan** |  |
| Test case |  | Commands | Expected Result |
| Push | 1 |  | 1 |
| 2 |  | 1, 2 |
| 3 |  | 1, 2,3 |
| 4 |  | 1,2,3,4 |
| 5 |  | 1,2,3,4,5 |

|  |  |  |
| --- | --- | --- |
| pop |  |  |
| pop |  |  |
| Peek () |  | 3 |
| isEmpty |  | FALSE |
| clear |  | isEmpty ==TRUE |

##### Lab Task 2 [ Link List Based Task]

Write code that takes a C++ code as input and validate the code for brackets. ({[]}) (Using Stack)

|  |  |  |
| --- | --- | --- |
|  | **Test Plan** |  |
| Test case | Commands | Expected Result |
| Validate | Test1.txt  #include<iostream> using namespace std;  int main () {  System(“Pause”);  } | Valid |
| Validate | Test2.txt  #include  <iostream> Using namespace std: Int main() {  {  Int arr[10]; for(int i=0;i<10;i++) cout<<i<<end; system(“pause”);  } | Valid |
| Validate | Test3.txt  #include<iostream> using namespace std;  int main()  {  for(int i=0;i<10;i++)  {  cout<<i<<end; system(“pause”);  } | Invalid |

##### Lab Task 3 [ Using Arrays]

Write a program that reads a line of text, changes each uppercase letter to lowercase, and places each letter both in a queue and onto a stack. The program should then verify whether the line of text is a palindrome (a set of letters or numbers that is the same whether read forward or backward)