**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Miss Arooj Khalil |
| Lab Instructor(s) | Mr. Dilawar Shabbir  Mr. Sohaib Ahmad |
| Section | SE-3A |
| Semester | Fall 2022 |

**Department of Computer Science**

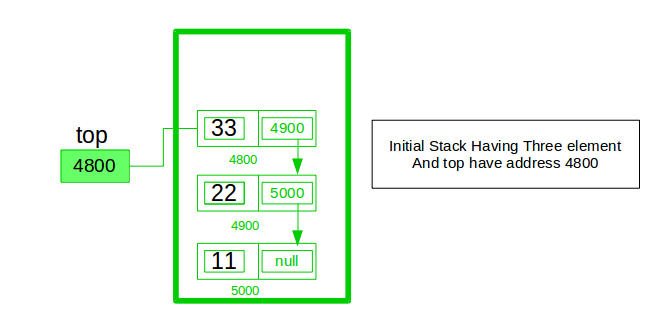
FAST-NU, Lahore, Pakistan

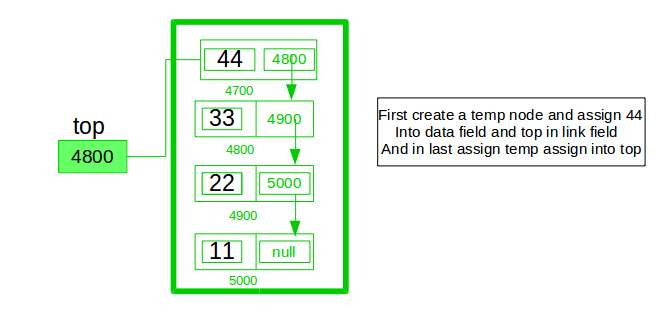
**Objectives:**

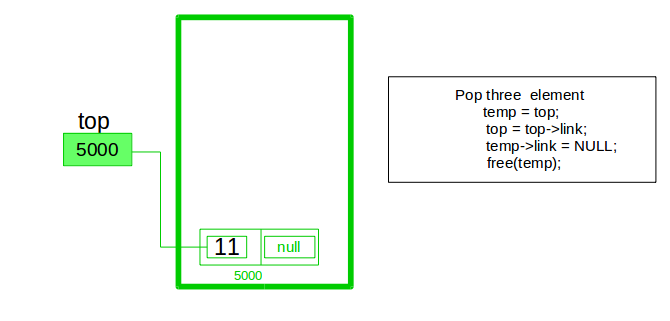
In this lab, students will practice:

1. Implementation of Stack.
2. Stack with single LinkedList.

**Implement a stack using singly linked list:** To implement a stack using singly linked list concept , all the singly linked list operations are performed 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. Using singly linked lists , we implement stack by storing the information in the form of nodes and we need to follow the stack rules and implement using singly linked list nodes . So, we need to follow a simple rule in the implementation of a stack which is last in first out and all the operations can be performed with the help of a top variable: **Pop , Push , Peek ,Display.**





In stack Implementation, a stack contains a top pointer. which is “head” of the stack where pushing and popping items happens at the head of the list. First node have null in link field and second node link have first node address in link field and so on and last node address in “top” pointer.  
The main advantage of using linked list over an arrays is that it is possible to implement a stack that can shrink or grow as much as needed. In using array will put a restriction to the maximum capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocated. so, overflow is not possible.

**Stack Operations:**

1. **push() :** Insert a new element into stack i.e just inserting a new element at the beginning of the linked list.
2. **pop() :** Return top element of the Stack i.e simply deleting the first element from the linked list.
3. **peek():** Return the top element.
4. **display():** Print all elements in Stack.

**Exercise No 1**

Implement a template-based stack using a singly linked list. The required member methods are:

1. **int size()**: returns the count of total element stored in the stack.
2. **bool isEmpty()**: returns true if the stack is empty else false.
3. **bool top(T&)**: returns, but does not delete, the topmost element from the stack via the parameter passed by reference. It returns false via a return statement if there is no element in the stack, else it returns true and assigns the top most element to the parameter passed by reference.
4. **void pop()**: deletes the top most element from the stack. If there is no element, return some error.
5. **push(T const& e)**: pushes the element “e” on top of the stack.

**Exercise No 2(a)**

Create a string using stack and remove all theoccurrences of the character (that is passed as an argument) from a string using Stack void removeAll(const T& char); // Function Prototype

Main() can look like this; however, you can change as per requirements.

int main()

{

stack<char> s;

s.push('v');

s.push('b');

s.push('r');

s.push('f');

s.push('b');

s.push('r');

s.removeAdjacent('r');

s.Display();

}

**Exercise No 2(b)**

Create a string using stack and remove all the triple adjacentoccurrences of the character (that is passed as an argument) from a string using Stack

void removeAdjacent (const T& char); // Function Prototype

Main() can look like this; however, you can change as per requirements.

int main()

{

Stack<char> s;

s.push('a');

s.push('a');

s.push('a');

s.push('b');

s.push('x');

s.push('x');

s.push('b');

s. removeAdjacent ('a');

s.display();

}

**Output:** bxxb

**Exercise No 3**

Delete middle element of a stack

Given a stack with push(), pop(), empty() operations, delete the middle of it without using any additional data structure.

**Input :** Stack[] = [1, 2, 3, 4, 5]

**Output :** Stack[] = [1, 2, 4, 5]

**Input :** Stack[] = [1, 2, 3, 4, 5, 6]

**Output :** Stack[] = [1, 2, 4, 5, 6]