# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2020** |
|  | | | | | |
| Points |  | Document author: | **Yet Chun Fong** | | |
|  | Author’s email: | **Fongyc7177@mail.fresnostate.edu** | | |
| Laboratory number: | **1** | | |
|  | | | | | |

**1. Statement of Objectives**

Introduce the elements of this experiment. Include a description of the objectives, scope, significance, and major accomplishments of this lab. Briefly explain what is covered in this report.

The elements of this lab session will be consisting of Stack and Linked List, which both are linear data structures. In this experiment, we covered the concepts for Stack and Linked list and understand the operations of these data structures. For Stack, the data that are inserted and deleted will be following the principle of last-in-first-out (LIFO) principle. We have to create 3 functions: Push, Pop, and Peek for the Stack class. In addition, we are required to create a variable named ‘top’ to keep track of the topmost value in the Stack, the ‘top’ variable is a major help when it comes to Pop and Peek function. However, we must always remember to update the position of variable ‘top’ in the Push and Pop function. Push is a function to insert a value on top of the Stack which makes the last inserted value be on the top on the Stack. Then, Pop is a function that remove the topmost value of the Stack. Then, Peek is a function that display the value on the top of the Stack. For Linked List, we are required to create a Node structure to store integer data and a pointer Next. Then, we have to create 3 functions, Insertion, Delete Front, and Delete Last. Similarly, we are required to create a variable ‘Head’ to keep stack of the frontmost value in the Linked List and move the variable along the Linked List when we traverse for more advanced insertion or deletion. This lab helped me to understand that the importance of Head pointer that always point to the first Node in a Linked List.

Going through the Coding Guidelines really helped me to understand the problems and think of the implementation of the functions before I started coding. Not only that, the examples for application of these data structures in real life situation helped me to realize that I am using Stacks and Linked List indirectly every day. These real-life examples are major helps in understanding the concepts behind data structures.

**2. Experimental Procedure**

List the procedure used in this lab. Include how you approached the question and why.

First, I looked at the Coding Guidelines provided by the Lab TA and starting to map out the implementation of functions for Stack and Linked List.

For Stack, I started off by defining ‘Max’ to 10, for the size of array for my Stack. Then I created a Class named Stack and declared an array value[Max], size of array = 10, I also declared an integer variable ‘top’ and set the value to -1 because the stack is empty before operation. The first function I created is Push, if the value of variable ‘top’ exceeded the size of 10, elements of (0-9), insertion will not be performed, but return to the main function. Else, ‘top’ will be incremented and position of the number in array will be following with the value of ‘top’. Second function I implemented is print function, I used a for loop to traverse through the Stack, the loop will be stopped if i is less than 0. The third function I implemented is Pop, first the function will check if the top is less than 0, as if top is less than 0 indicates that the Stack is empty, and nothing can be Pop. Else, the number at position value[top] will be removed and ‘top’ will be decrement, because ‘top’ will be change to the next topmost number after the removal of ‘top’ value. The last function I implemented is Peek, which it will check the value of ‘top’, if ‘top’ is less than 0, return to main function. Else, a temporary variable named ‘x’ will be created to store the number of values at position ‘top’ then display the number.

For Linked List, I started off by creating a structure Node and integer data variable to store integer data and a pointer variable that points to the next Node. Then I created a Class named List with a pointer of ‘head’ to point to the front Node in the Linked List as a private variable in the Class. I also included a constructor. The first function I implemented is Print, I declared a new pointer that point to the head and used a while loop with condition of h!=NULL and print out the elements in linked list. The second function I implemented is an insertion function which insert to the front of the linked list. The function will create a new Node that points to the data to store number inserted while calling the function and point next to the head of initial Linked List. After that, the pointer ‘head’ will be updated and assigned to the new Node. Third function I implemented is deleteFront, which it will delete the first node of the linked list. The function will start off by checking if the head is NULL which indicates that the Linked List is empty. Else, a temporary Node will be used to point to head, and I move head to the next Node and delete the temporary Node that used to point to the head. Moving to the last function in the List Class, deleteLast, this function is used to delete the last Node in the linked list. I have created a temporary Node that points to head (first node in the linked list) and another new pointer named ‘prev’, so that I can keep track of the previous Node of head. An ‘if statement’ is used to check if the head is NULL which indicates that the linked list is empty. Then another ‘if statement’ is initialized to check if the next node of head is NULL, if the next node of head is NULL, means that the head is last node. Thus, head will be assigned to NULL and delete the temporary node that used to point to head because we still need to keep the head for insertion. Then, a while loop is used to traverse the linked list until the next of head node is NULL, which means that the head node is the last node. I used the ‘prev’ here so that I can keep track of the second last node and delete the last node using prev->next = NULL.

**3. Analysis**

Discuss the experimental results. Include the screenshots of the results.

A screenshot of a cell phone

Description automatically generated

First, items in the stack are 60, 40 and 20. Then I executed Pop function to remove 60 from the Stack, and also executed Peek function to see what’s the topmost value in the Stack which is 40 because I popped 60. Then, I print again, the items in the stack are only 40 and 20.

A screenshot of a cell phone

Description automatically generated

Firstly, the items in the Linked List are 20,40, and 60. Then I executed deleteFront function which delete the first node in the Linked List, 20. And I also executed deleteLast function which delete the last node in the Linked List, 60. Therefore, the Linked List only contains 40 at last.

**4. Encountered Problems**

Describe the issues you faced and how you tackled them. Also, you can explain if you could not solve the issue. You should also include errors and discrepancies.

For Stack, I need to keep reminding myself to update the value of ‘top’ when Push and Pop. At first, I forgot to perform increment for ‘top’ and the Peek function is not working properly. Then, only I realized that I forgot to include top++ in my Push Function. Secondly, I forgot to check the current state of Stack before I perform Push operation, then it might have the problem of Stack Overflow. Therefore, I fixed the problem by by including an if statement to check the value of ‘top’, see if it exceeded Max-1. For the Pop, I know that I need to check for Stack Underflow, so I have an if statement in the first place to check for Stack Underflow by checking the value of ‘top’.

For Linked List, I was having data type error in the insertFront function. As I assigned head to the new Node ‘a’, I am having compilation error saying that the system could not convert Node datatype to integer datatype. After I did a round of checkup on my code, my ‘head’ Node and ‘a’ Node is the same datatype of Node. So, I compile it again, and it works fine. Therefore, I am not too sure what the problem is. Secondly, I need to keep telling myself not to delete or modify the ‘head’ pointer because if I accidentally deleted the head pointer when I execute deleteFront or deleteLast, the Linked List that I created might be having errors. To avoid this problem, I always declare a new pointer that points to the ‘head’ pointer. If I want to delete the ‘head’ pointer, I can just change the head to the next node and delete the new pointer that is assigned to the head pointer.

**5. Conclusions**

Summarize your conclusions with a list of what you learnt in this lab.

* Initiate a variable ‘top’ to keep track of the topmost item in Stack
* Push and Pop of Stack is based on the principle of last-in-first-out (LIFO)
* Always check the current state of Stack before Push or Pop (Check for Stack overflow/ underflow)
* Increment of ‘top’ during Push operation / Decrement of ‘top’ during Pop operation
* Only print topmost value during Peek operation, do not pop it.
* For linked list, always remember to reassign the head after deletion.
* Always check the current state of head node before any operation (Check if it’s NULL)
* For deletion, always create a temporary node to point to the head node then delete the temporary node.
* Make sure that the datatypes of Node are the same, if not there will be compilation error.

**6. References**

List the references used in this report.

* Personal notes from CSCI 41
* Coding Guidelines on Canvas