# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Spring 2022** |
|  | | | | | |
| Points |  | Document author: | **MingkuanPang** | | |
|  | Author’s email: | **Yafking20 @mail.fresnostate.edu email** | | |
| Laboratory number: | **03** | | |
|  | | | | | |

**1. Statement of Objectives**

This lab asks for implementing two data structures: stack and linked list. For stack, we have push, pop, and peek operations. For linked list, we have insert and delete operations. In this report, I will describe the procedures that I wrote in the program and the way I implemented them.

**2. Experimental Procedure**

**For stack:**

Text

Description automatically generated

**Top:**

I initialized the top to the value of -1, because the first index of an array is 0, and that is when the stack is empty.

**Push:**

For the push procedure, it takes an integer, and it will judge if the stack is already full, if it is not, then it will put the number into array, and meanwhile top = top +1. In that way, we keep the value of top to be the index of last element in the array. Otherwise, it will throw an error.

**Pop:**

For the pop procedure, it will judge if the stack is empty, if it is not then it will minus top with 1: top = top -1, then return array[top+1]. Otherwise, it will throw an error.

**Peek:**

For the peek procedure, it will judge if the stack is empty, if it is not then it will return array[top].

Otherwise, it will exit the program.

**Display:**

For display procedure, it will print the elements of the array with reverse order.

**For node:**

Graphical user interface, text

Description automatically generated

To build a linked list, I must make a node first. There are two elements in a node. One is its value; another one is a pointer pointing to next node. I also created a default constructor to make it easier to be initialized.

**For linked list:**

Text

Description automatically generated

**Insert:**

For the insert procedure, I basically make the new element to be a new node, and then make the head to be its next node, finally move the head pointer to the new node.

**Delete\_begin:**

For the delete\_begin procedure, I assigned head node to a temp node, then I moved the head pointer to the next node. Finally, I remove the temp node to free the memory.

**Delete\_end:**

For the delete\_end procedure, I searched the second node from the back first, and then assigned its next node to a temp node, then make its next node to NULL. Finally delete the temp node to free the memory.

**3. Analysis**

**Stack:**

Input 1 (basic operation of the stack):

Text

Description automatically generated

Output 1:

Graphical user interface, text, chat or text message

Description automatically generated

Input 1 (push function will determine if the stack is full before trying to add an element)

Graphical user interface, text

Description automatically generated

Output 1:

Text

Description automatically generated

Input3 (pop will determine if the stack is empty before popping an element out):

Text

Description automatically generated

Output 3:

Graphical user interface

Description automatically generated

**Linked list:**

Input 1 (basic operation of linked list):

Text

Description automatically generated

Output (For the linked list basic operations):

Text

Description automatically generated

Input (For the linked list basic operations without any node inserted):

A screenshot of a computer

Description automatically generated

Output (For the linked list basic operations without any node inserted):

Graphical user interface, text

Description automatically generated

**4. Encountered Problems**

When I tried to deleted node in an empty linked list, my compiler raised an error. After while I realized that is because I tried to get next node from a NULL node, that does not make any sense. Therefore, I should determine if a node is NULL before I try to get its value or its next node.

Before:

Graphical user interface, text, application, chat or text message

Description automatically generated

After:

Text

Description automatically generated

**5. Conclusions**

We should think deeply before we start writing codes, that way we can avoid many bugs. Before we try to operate data structures, we need to think about different cases. Such as if we try to get an element from an array, but what if the array is empty? Therefore, we should examine these cases to ensure the safety of our codes.

**6. References**

I did not use any references in this report.