New York University Abu Dhabi — Department of Engineering

Data Structures and Algorithms
ENGR-UH 3510

Pre Lab 2

Basic Data Structures

Stack

Theory

Each record of a linked list is often called an 'element' or 'node'. The field of each node that contains the address of the next node is usually called the 'next link' or 'next pointer'. The remaining fields are known as the 'data', 'information', 'value', 'cargo', or 'payload' fields.

Singly linked lists contain nodes which have a data field as well as 'next' field, which points to the next node in line of nodes. Operations that can be performed on singly linked lists include insertion, deletion and traversal. The 'head' of a list is its first node. The 'tail' of a list may refer either to the rest of the list after the head, or to the last node in the list.[1]

Here is a summary of the structure of a singly-linked list and the various operations that can applied onto it.

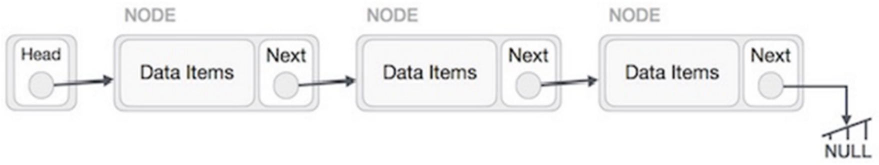

Figure 1: Single Linked List Structure

As per the above illustration, following are the important points to be considered:

- Head: Only contains the pointer called Next
- Node: Each link (edge) carries a data field(s) and a pointer called Next
- Each node is linked with its Next pointer
- Tail node: Its Next pointer points to NULL

A stack is a basic data structure that can be logically thought of as a linear structure represented by a real physical stack or pile, a structure where insertion and deletion of items takes place at one end called top of the stack. The basic concept can be illustrated by thinking of your data set as a stack of plates or books where you can only take the top item off the stack in order to remove things from it. This structure is used all throughout programming.

The basic implementation of a stack is also called a LIFO (Last In First Out) to demonstrate the way it accesses data, since as we will see there are various variations of stack implementations.
There are basically three operations that can be performed on stacks.[2]

1. inserting an item into a stack (push).
2. deleting an item from the stack (pop).
3. displaying the contents of the stack(print).

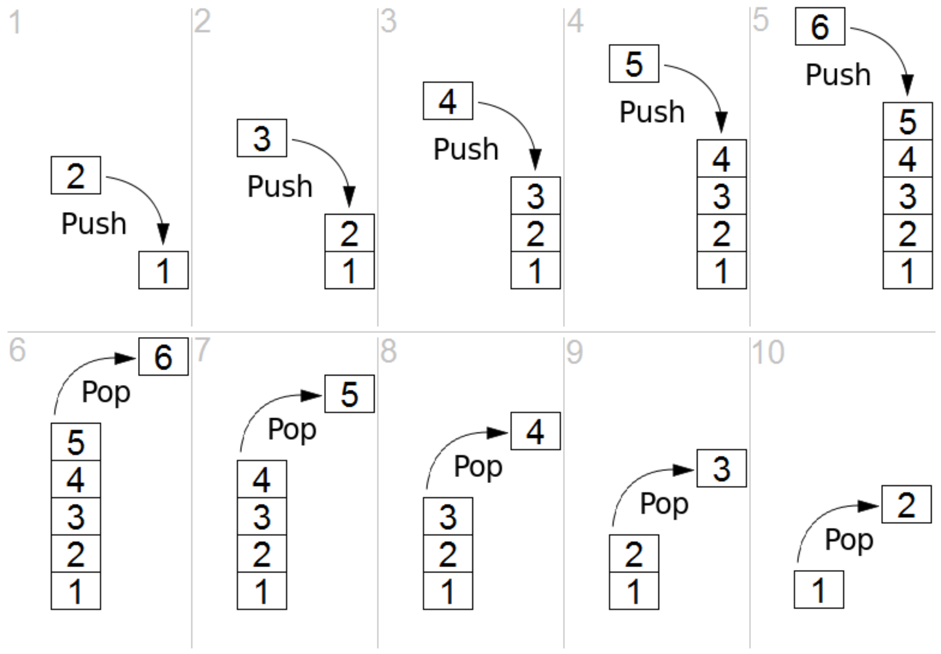

Figure 2: Stack and its Operations

Assignment

Implement a **dynamic stack** using a **single linked list** as basis where every node is interconnected to the next one.

1. Define the struct that is needed following the below definition, if you assume that you need additional attributes you may add them.

```
Node {
    int data;
    Node next;
}
```

Listing 1: Node class

```
Queue {
    Node head;
}
```

Listing 2: Queue class

2. Implement a push function that enables insertion on new items to the stack.
3. Implement a pop function that enables removal of already existing items from the stack.
4. Implement a print functions that displays the contents of the stack.
5. Implement a main functions that demonstrates the aforementioned functionality. It is up to you to define the sequence of operations that need to be done in order to better demonstrate your code. Always use the needed print outs.

References

[1] Wikipedia. *Singly linked list.* URL: https://en.wikipedia.org/wiki/Linked_list#Singly_linked_list.

[2] Wikipedia. *Stack (abstract data type).* URL: https://en.wikipedia.org/wiki/Stack_(abstract_data_type).