CS2023 - Data Structures and Algorithms
In-class Lab Exercise

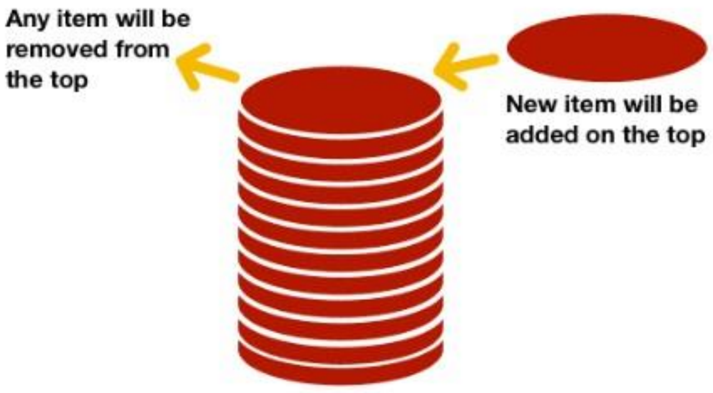
Week 6

You are required to answer the below questions and submit a PDF to the submission link provided under this week lab section before end of the session time (no extensions will be provided). You can either write / type your answers, but either way your answers should be readable.

Create GitHub repository, add your codes there and add respective link to the submission file.

Exercise:

Stack is a linear data structure which is a collection of elements which are inserted or deleted according to the LIFO rule i.e. Last-In-First-Out. Take the example of a stack of disks which are placed one on top of another we keep on adding new disks to the top and when we need to take one, we take the topmost one very similar to how a stack works.


Any item will be removed from the top
New item will be added on the top

Operations performed with a Stack:

- Push() - To insert data into the stack
- Pop() - To remove/delete data from the stack
- isEmpty() - To check whethet a stack is empty or not
- isFull() - To check whethet a stack is full or not
- StackTop() - To find what is at the top of the stack
- Display() – To print elements in the stack

1. Implement Stack and its functions using **Array.** Consider the given example pseudo code for push and pop using array.
   Watch the following video for more details:
   https://www.youtube.com/watch?v=rS-ZKTqwi90&ab_channel=NesoAcademy

```
PUSH(S, x)
  S.top = S.top+1
  if S.top > S.size
      Error "Stack Overflow"
  else
      S[S.top] = x
```

```
POP(S)
  if IS_EMPTY(S)
      Error “Stack Underflow”
  else
      S.top = S.top-1
      return S[S.top+1]
```

2. Implement Stack and its functions using **LinkedList**. Consider the given example pseudo code for push and pop using linked list.
   Watch the following video for more details:
   https://www.youtube.com/watch?v=0-kkDfCOXOI&ab_channel=NesoAcademy

```
PUSH(S, n)
  if IS_EMPTY(S) //stack is empty
      S.head = n //new node is the head of the linked list
      S.top = n //new node is the also the top
  else
      S.top.next = n
      S.top = n
```

```
POP(S)
  if IS_EMPTY(S)
       Error "Stack Underflow"
  else
       x = S.top.data
       if S.top == S.head //only one node
            S.top = NULL
            S.head = NULL
       else
            tmp = S.head
            while tmp.next != S.top //iterating to the node previous to top
                 tmp = tmp.next
            tmp.next = NULL //making the next of the node null
            S.top = tmp //changing the top pointer
       return x
```

3. Execute the following operations if Stack. Compare the time taken for execution between your implementation using array and LinkedList. (Note: you can randomize the value for push operation)

Push(8)
Push(10)
Push(5)
Push(11)
Push(15)
Push(23)
Push(6)
Push(18)
Push(20)
Push(17)
Display()
Pop() $\times$ 5 times
Display()
Push(4)
Push(30)
Push(3)
Push(1)
Display()

Results:

Test No.	Array Approach Time	Linked List Approach Time
1	3825900	1684300
2	4930500	1696400
3	4889000	1743600
4	4526200	1575500
5	4758700	1749600
average	4580060	1695880

Comparison: Array implementation have taken more time compared to linked list implementation. But linked list implementation takes more space in memory.

Git repo link : https://github.com/Hansa2000/CS2023-labs