CSCI 115 Lab

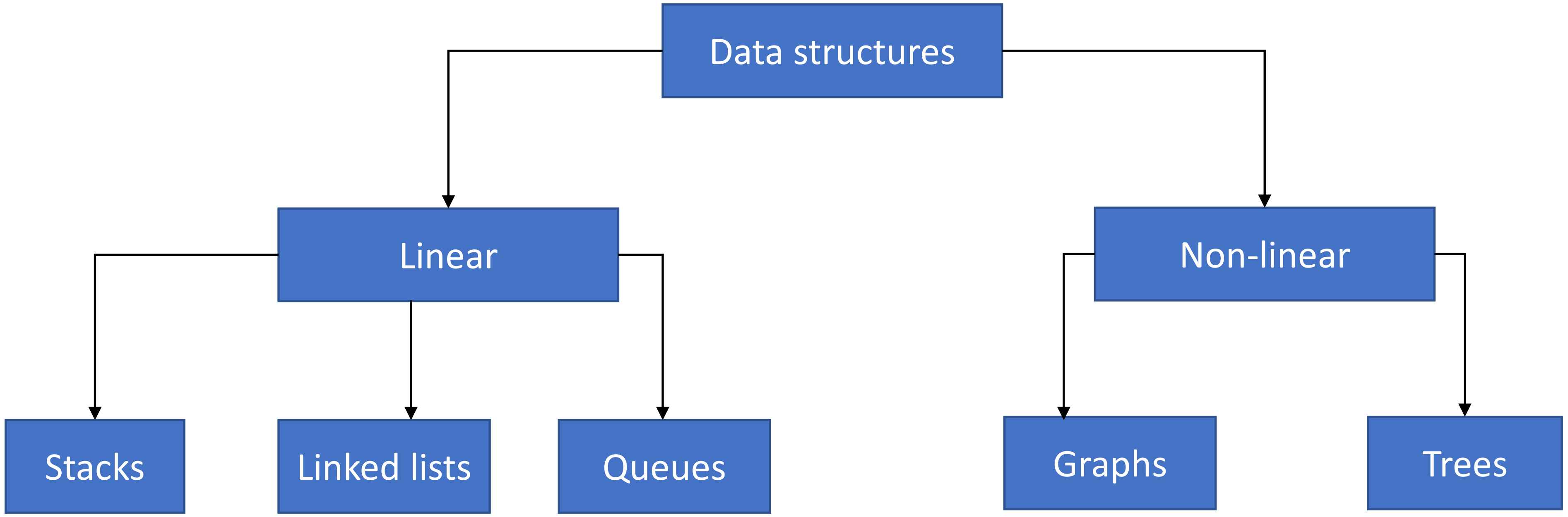
Week 2- Stacks and Linked List

- Professor: Dr. Matin Pirouz
  Email: mpirouz@csufresno.edu

- TA: Shreeja Miyyar
  Email:shreejarao12@mail.fresnostate.edu

Table of Contents

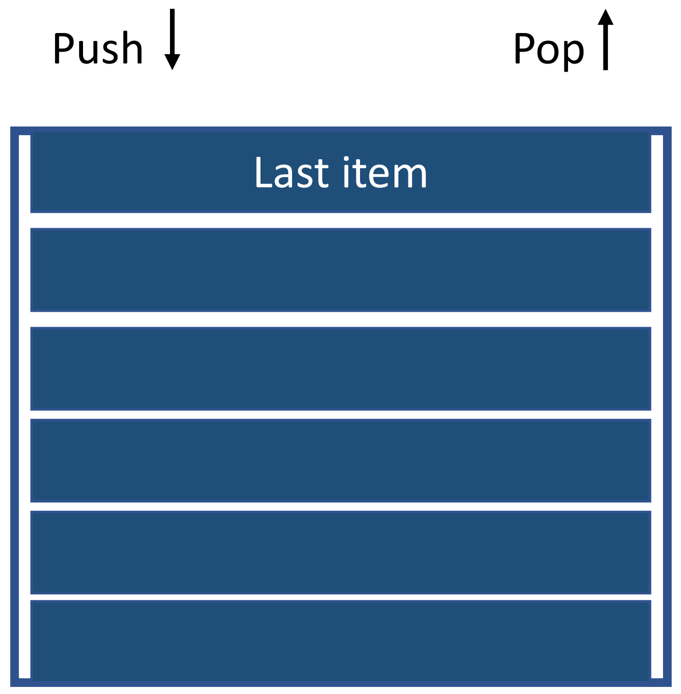
Data Structure Classification



Data Structures are the programmatic way of storing data so that data can be used efficiently.

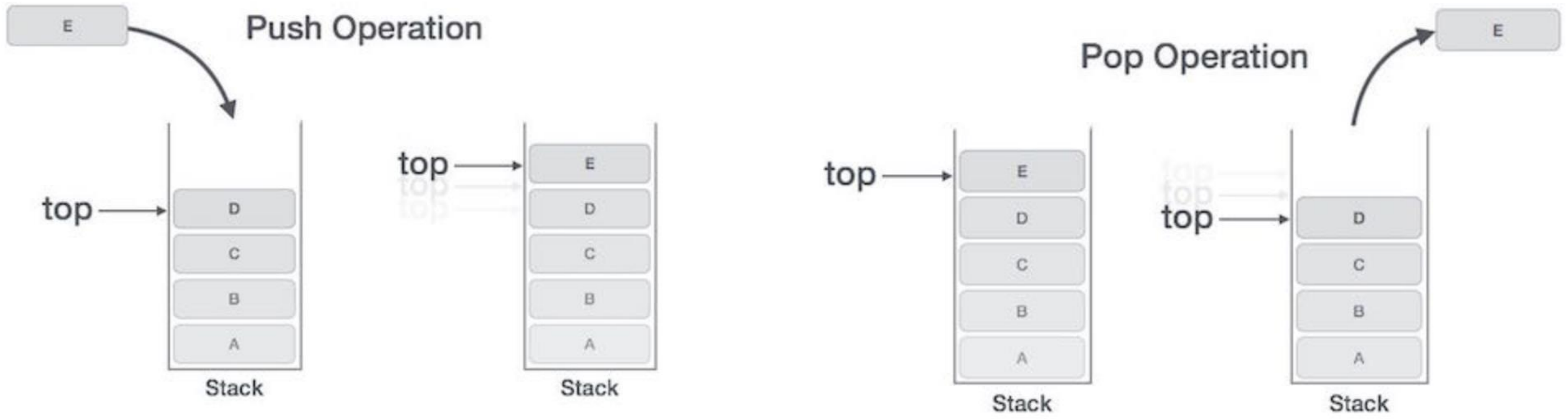
Stacks

- A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle.
- LIFO - the last item which is pushed, is the first item which is popped.



Stack operations

- PUSH
- POP
- PEEK (TOP)



PUSH Operation

- **Step 1** − Checks if the stack is full.
- **Step 2** − If the stack is full, throw an error and exit.
- **Step 3** − If the stack is not full, increment the “top” to point to the next empty space and insert item in that position.
- **Step 4** − Returns success.

POP Operation

- **Step 1** – Checks if the stack is empty.
- **Step 2** – If the stack is empty, throw an error and exit.
- **Step 3** – If the stack is not empty, remove the top item from the array and return it.
- **Step 4** – Decrease the value of top by 1.
- **Step 5** – Return success.

PEEK (Top) Operation

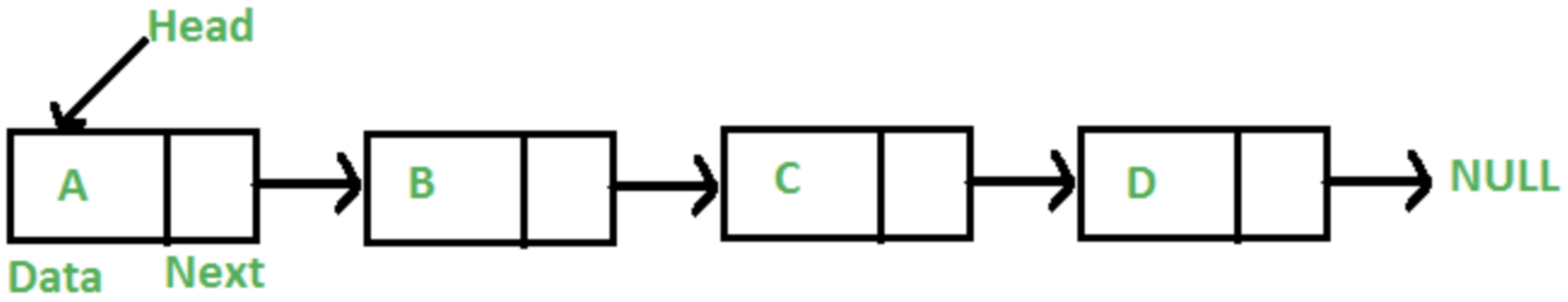
- **Step 1** − Checks if the stack is empty.
- **Step 2** − If the stack is empty, throw an error and exit.
- **Step 3** − If the stack is not empty, return the top item of the array
- **Step 4** − Returns success.

Applications of stacks

- String reversal
- Validity of nested parenthesis: For each opening parenthesis there is a corresponding closing parenthesis.
  - Example
    - (()) -> valid
    - (()(-> invalid
    - ()() -> valid
- Undo (Ctrl/CMD + Z)

Linked Lists

- It is a linear data structure containing nodes
- Nodes are structures made up of data and a pointer.
- The pointer in a node points to the next node.
- The first node is generally called the HEAD node.
- The last node will always point to NULL, indicating end of linked list sequence.

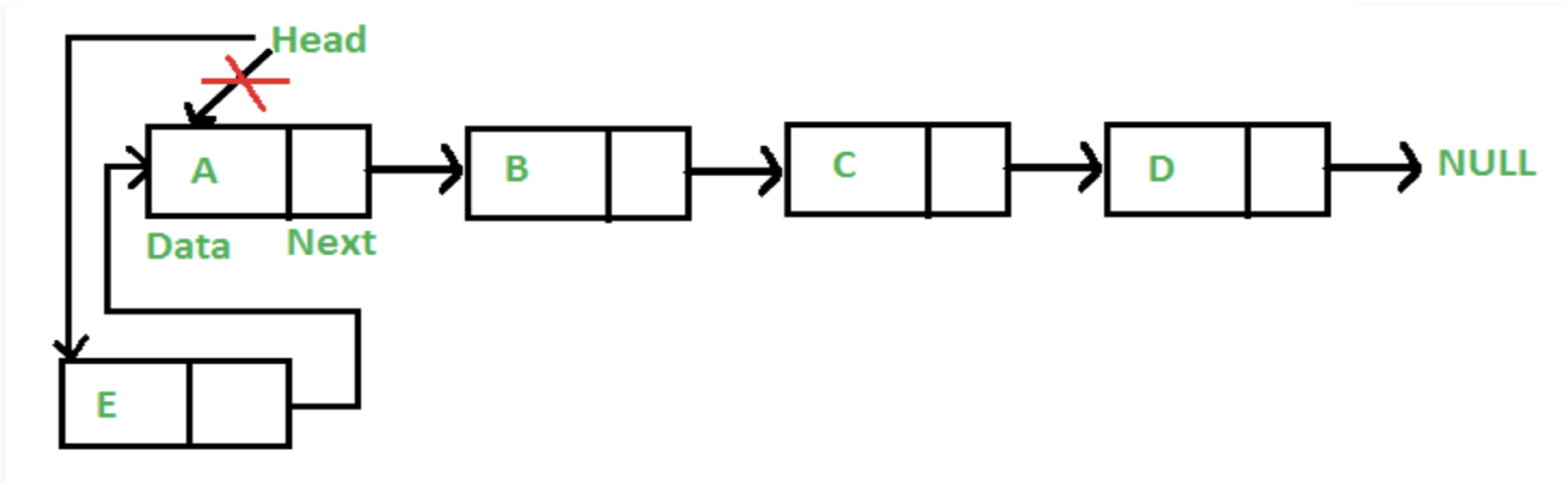


Linked List operations

- **Insertion** − Adds an element at the beginning of the list.
- **Deletion** − Deletes an element at the beginning/end of the list.
- **Display** − Displays the complete list.

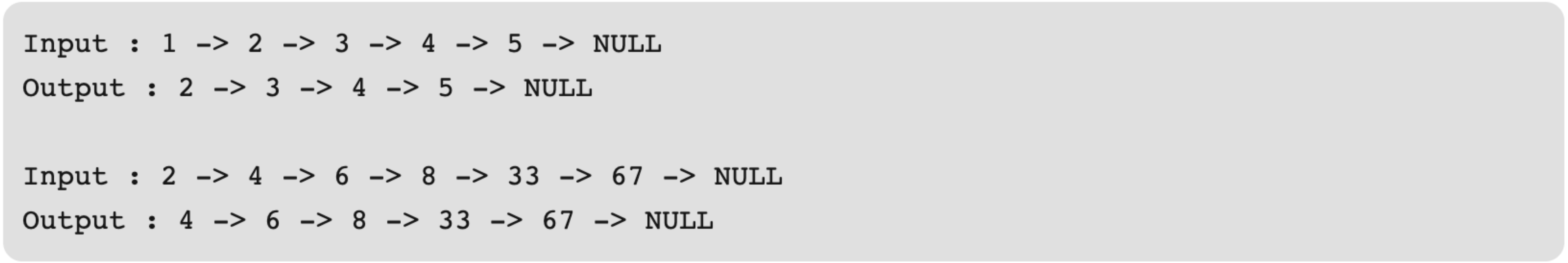
Insert operation (at the beginning)

- Create a new node with data
- Point the next of the new node to HEAD.
- Move the HEAD to point to the new node



Delete operation (at the beginning)

- Store the HEAD node in a temporary node
- Move the HEAD node to point to the next node
- Delete the temporary node

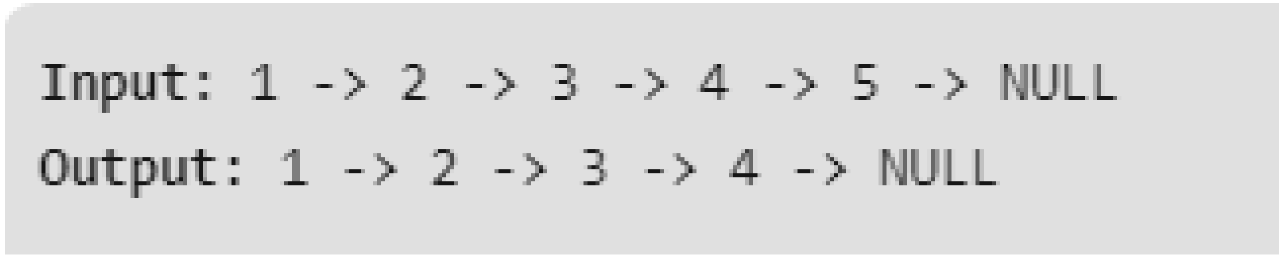


```
Input : 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Output : 2 -> 3 -> 4 -> 5 -> NULL

Input : 2 -> 4 -> 6 -> 8 -> 33 -> 67 -> NULL
Output : 4 -> 6 -> 8 -> 33 -> 67 -> NULL
```

Delete operation (at the end)

- If head node is null or if there is only one node then return null.
- Traverse the linked list till the second last node is reached.
- Delete the last node and change the next value of second last node to null.



```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Output: 1 -> 2 -> 3 -> 4 -> NULL
```

Applications of Linked Lists

- Implement stacks and queues
- Dynamic memory allocation
- Performing arithmetic operations on long integers - Each integer is stored in a Node.

Lab Assignment

1) Write a program to create a stack data structure (do not use the inbuilt stack class) using arrays. It should support Push, Pop and Peek operations.

To-do:

- Create a stack using arrays with maximum array size 10.
  Hint: Define an empty array of type integer with size 10

- Push integers 20, 40, 60 into the stack.
  Hint: Create a class Stack with variable defined to track topmost element of stack (top) and define a method push which accepts an integer as an argument and inserts the argument into the array and increments the top by 1.

- Print the items in stack.
  Hint: Loop through the array till top and print the items.

- Pop the item from the stack and print the popped item.
  Hint: Add a method pop in Stack class and decrement the variable top.

- Peek the stack and print the top-most element of the stack (do not remove it)
  Hint: Return the topmost element of the array without decrementing top.

- Print the stack.
  Hint: Loop through the array till top and print the items.

Coding Guidelines

- Start coding by creating a class Stack which has a variable *top* and an array of size 10. Add 4 methods to Push, Pop, Peek and Print.
- Inside the Push method, check if the top is greater than size of the array. If yes, it represents Stack Overflow. If not, perform insertion.
- Inside the Pop & Peek method, check if top is less than 0. If yes, it is Stack Underflow. If not, perform pop & peek, respectively.
- Inside Print method, loop through all elements in the array till top and print them.
- In main function, create an object of Stack class and call the required methods.

2) Write a program to create a linked list structure. It should support insert and delete operation.

To-do:

- Insert a node with data value 60 in the first position
  Insert a node with data value 40 in the first position.
  Insert a node with data value 20 in the first position.
  20 -> 40 -> 60

  Hint: Create a struct for node with variables data (int) and pointer (struct node type which keeps track of the next node). Write an insert method which receives the data value as an argument and creates a node with that data and inserts it in the first position as described in the algorithm.

- Print the linked list elements
  Hint: Traverse the linked list till it reaches null and print the data.

- Delete the first element of the linked list and print the linked list
  40 -> 60
  Hint: Create a temporary node to store the head and then point head to the second element.

- Delete the last element of the linked list and print the linked list
  40
  Hint: Traverse till last but one node in the linked list and set the pointer of that node to null

Coding Guidelines

- Start coding by creating a structure Node which has a data variable & pointer variable which points to the next node.
- Define head node pointer and assign it to null.
- Define a methods insert, deleteFront, deleteEnd and print.
- Remember to check if head is null. If yes, delete operation cannot be performed.
- Create a main function and call the required methods.

Questions?