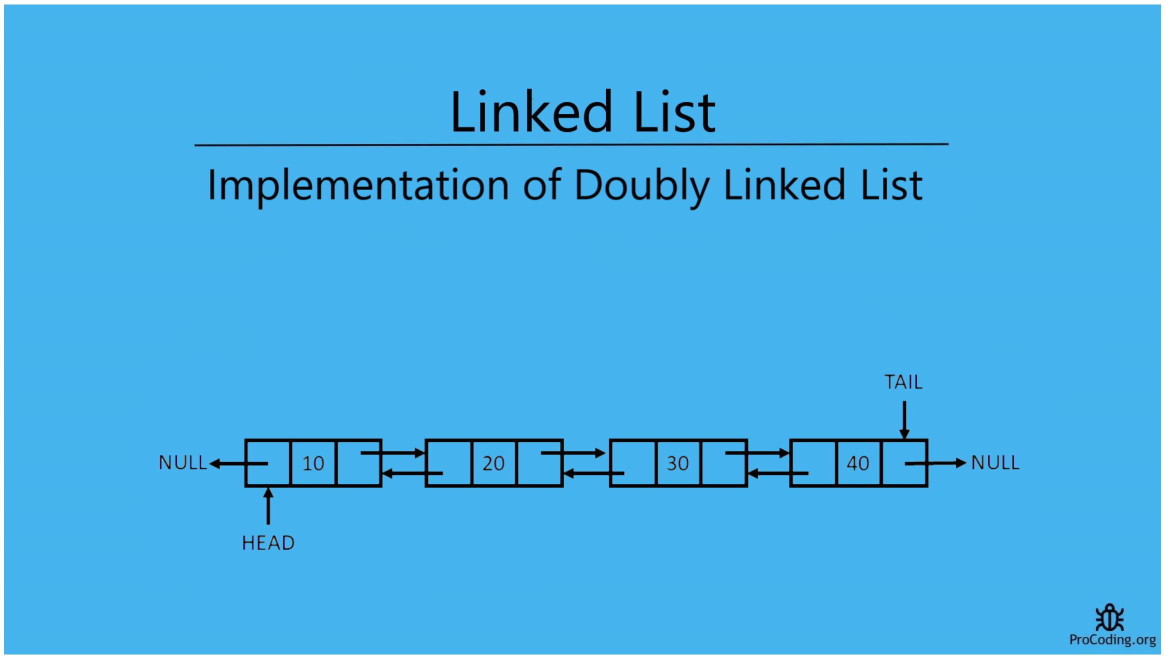
# Data Structures Fall 2023

# Lab 07: Doubly & Circular Linked List

## Doubly Linked List:

A doubly linked list is a data structure used for organizing and storing a collection of elements, such as integers, strings, or custom objects. It is similar to a singly linked list but with an important difference: each node in a doubly linked list contains not only a reference to the next node in the list (as in a singly linked list) but also a reference to the previous node. This dual reference to both the next and previous nodes allows for more efficient traversal in both directions, forward and backward.



A single node is also pretty simple. It has three parts: data, or the information that the node contains, a reference to the next node, and a reference to a previous node.

## Circular Linked List:

A circular linked list is a variation of a linked list in which the last node points back to the first node, creating a loop or circle. In other words, the next pointer of the last node points to the first node, forming a closed-loop structure.

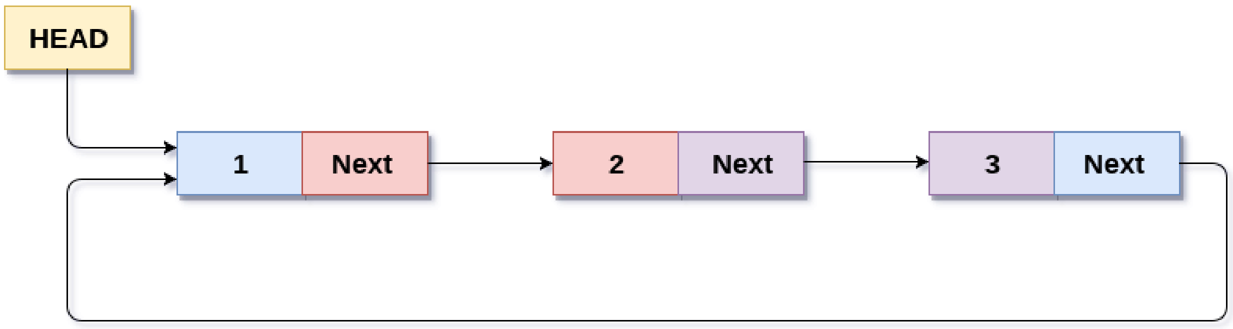
Key characteristics of a circular linked list:

1. Circular Structure: Unlike a regular linked list, where the last node's next pointer is typically null, in a circular linked list, the last node's next pointer points back to the first node.

2. Continuous Loop: You can traverse the entire list starting from any node and keep going until you reach the starting node again, making it useful for certain applications like scheduling and circular buffers.

3. No End Sentinel: There's no distinct "end" or "null" element in a circular linked list; instead, the loop structure signifies the end of the list.

Circular linked lists are often used in situations where you need continuous access to elements and want to avoid the need to reach the end of the list explicitly. They can be helpful for implementing algorithms that involve circular or repeating processes.



Circular Singly Linked List

## Task01

Design a doubly linked list class with the following specifications:

1. **DoublyNode Class:**
   - Create a class called `DoublyNode` with the following data members:
     - int data to store the data for the node.
     - DoublyNode* prev to store a pointer to the previous node.
     - DoublyNode* next to store a pointer to the next node.

   - Provide the following member functions:
     - A default constructor for `DoublyNode`.
     - A parameterized constructor that sets the data, prev, and next pointers.
     - Getter methods for data, prev, and next.
     - Setter methods for data, prev, and next.

2. **DoublyLinkedList Class:**
   - Create a class called `DoublyLinkedList` to manage a doubly linked list. It should have a pointer to the head node initially.
   - Provide the following member functions:
     1. A default constructor for DoublyLinkedList.
     2. A function called `insert` to insert a data item at the end of the doubly linked list. If the list is empty, the new node becomes the head.
     3. A function called insertToHead to insert a data element at the beginning of the doubly linked list.
     4. A function called isEmpty to check if the doubly linked list is empty.
     5. Explain why there is no need to implement an `isFull` function.
     6. A function called `search` to search for a specific element provided by the user and return true if found, and false otherwise.

7. A function called `update` to update/replace one data element with another.
8. A function called `insertAtIndex` will count the data elements in the doubly linked list and place a new data element in the specified location (indexed by counting from 0).
9. A function called `deleteData` to delete a specified data element. This should involve updating the previous node's `next` pointer and the next node's `prev` pointer accordingly.
10. A function called `print` to print the data of the doubly linked list from the head to the end.

Your task is to implement both the `DoublyNode` and `DoublyLinkedList` classes based on the provided specifications, and ensure that they work correctly as described.

-----------------------------------------------------------------------------------

## Task2

Design a circular linked list class with the following specifications:

**1. Node Class:**

- Create a class called `Node` with the following data members:
  - `int data` to store the data for the node.
  - `Node* next` to store a pointer to the next node.

- Provide the following member functions:
  - A default constructor for `Node`.
  - A parameterized constructor that sets the data and next pointer.
  - Getter methods for data and next.
  - Setter methods for data and next.

**2. CircularLinkedList Class:**

- Create a class called `CircularLinkedList` to manage a circular linked list. It should have a pointer to the head node initially.

- Provide the following member functions:
    1. A default constructor for `CircularLinkedList`.
    2. A function called `insert` to insert a data item at the end of the circular linked list. If the list is empty, the new node becomes the head, and it should point to itself.
    3. A function called `isEmpty` to check if the circular linked list is empty.
    4. Explain why there is no need to implement an `isFull` function.
    5. A function called `search` to search for a specific element provided by the user and return true if found, false otherwise.
    6. A function called `update` to update/replace one data element with another.
    7. A function called `insertAtIndex` that will count the data elements in the circular linked list and place a new data element in the specified location (indexed by counting from 0).
    8. A function called `deleteData` to delete a specified data element.
    9. A function called `print` to print the data of the circular linked list starting from the head node and continuing until it reaches the head node again.

Your task is to implement both the `Node` and `CircularLinkedList` classes based on the provided specifications and ensure that they work correctly as described.

---------------------------------------------------------------------------------

## Task 3:

There are N people, numbered from 1 to N, standing in a circle. Each person has a unique skill level represented by an integer value. Starting from person 1, a selection process is followed. After every M selection, the person with the lowest skill level is eliminated. The process continues until only one person remains. The last remaining person is declared the winner. Your task is to write a program to find the winner's skill level.

Write a function **findWinner()** that uses the classes created in the previous question that implements this functionality. Also, display minimum skill in each case.

| Input | Output |
|---|---|
| M=2, N=6 | 6 |
| M=3, N=9 | 9 |