**1. Node Structure**

* Defines a structure **Node** with members **PRN** (Personal Registration Number), **name**, and a pointer **next** to the next node.

**2. PinnacleClub Class**

* Represents a club with members stored as linked nodes.
* Contains private members **president** and **secretary**, both of type **Node\***, representing the head and tail of the linked list.

2.1 Constructor

* Initializes **president** and **secretary** to **nullptr** in the constructor.

2.2 **addMember** Method

* Adds a new member to the club with the specified **prn** and **name**.
* Creates a new node, and if the club is empty, sets both **president** and **secretary** to the new node.
* Otherwise, appends the new node to the end of the list and updates **secretary** to point to the new node.

2.3 **deleteMember** Method

* Deletes a member with the specified **prn**.
* Traverses the linked list to find the member, deletes the node, and adjusts the pointers accordingly.

2.4 **displayMembers** Method

* Displays the details of all club members by traversing the linked list.

2.5 **totalMembers** Method

* Counts and returns the total number of members in the club.

2.6 **concatenateLists** Method

* Concatenates the members of another **PinnacleClub** (**otherClub**) to the current club.
* If the current club is empty, sets its head and tail to the head and tail of the other club.
* Otherwise, appends the members of the other club to the end of the current club.

**3. main Function**

* Creates two instances of **PinnacleClub** named **division1** and **division2**.
* Adds members to each division using the **addMember** method.
* Displays the members of each division using the **displayMembers** method.
* Concatenates the lists of **division2** to **division1** using the **concatenateLists** method.
* Displays the concatenated list using **displayMembers**.

**Note:**

* The program demonstrates a basic linked list implementation in a club scenario.
* The **Node** structure represents individual members, and the **PinnacleClub** class manages the list of members with various methods.
* The **concatenateLists** method is used to combine the members of one club with another.
* The program uses dynamic memory allocation (**new** and **delete**) to manage nodes in the linked list.
* The member data includes a PRN and name, but additional data could be added as needed.

Algorithm:

1. \*Initialization:\*

- Define a constant SIZE for the size of the queue.

- Declare an integer array Queue of size SIZE to implement the queue.

- Initialize front and rear to -1, indicating an empty queue.

2. \*isEmpty Function:\*

- Check if the queue is empty by comparing front and rear.

- If front is one less than rear, the queue is empty.

3. \*isFull Function:\*

- Check if the queue is full by using modulo operation on (rear + 1) % SIZE.

- If the result is equal to front, the queue is full.

4. \*enque Function:\*

- Check if the queue is not full using Qfull() function.

- If the queue is initially empty, set front to 0.

- Increment rear and insert the data into the Queue.

5. \*deque Function:\*

- Check if the queue is not empty using isEmpty() function.

- Retrieve the element at the front of the queue (Queue[front]).

- Increment front to dequeue the element.

6. \*Main Function:\*

- Perform enqueue and dequeue operations to demonstrate the functionality of the queue.

Top of Form