1. **Node Class:**
   * Defines a **Node** class to represent a node in a doubly linked list.
   * Each node has a boolean **bit**, a **next\_node** pointer, and a **prev\_node** pointer.
   * The **bit** attribute represents the binary digit (0 or 1).
2. **List Class:**
   * Defines a **List** class to represent a linked list of binary digits.
   * Includes private members **start** and **last**, which point to the first and last nodes in the linked list.
3. **List Methods:**
   * **input() Method:**
     + Takes a binary string as input from the user and creates a linked list by converting each character to a binary digit.
     + Each digit is represented by a node in the linked list.
   * **one\_complement() Method:**
     + Inverts (flips) each bit in the linked list to obtain the one's complement.
   * **two\_complement() Method:**
     + Calls **one\_complement()** to obtain the one's complement.
     + Adds 1 to the one's complement to get the two's complement.
   * **print() Method:**
     + Prints the binary representation of the linked list by traversing the list from **start** to **last**.
   * **add\_bit\_at\_begin(bool x) Method:**
     + Adds a new node with the specified bit **x** at the beginning of the linked list.
     + Used for carrying in the addition operation.
   * **Overloaded + Operator (operator +()) Method:**
     + Overloads the **+** operator to perform binary addition of two linked lists.
     + Adds two binary numbers represented by different linked lists.
4. **main Function:**
   * Creates three instances of the **List** class (**l**, **l1**, and **l2**).
   * Enters a loop where the user can choose to perform operations on binary numbers:
     + One's complement
     + Two's complement
     + Addition of two binary numbers
   * The user can continue trying different operations until they choose to exit.
5. **User Interaction:**
   * The program interacts with the user by taking binary numbers as input and providing options to perform various operations.
   * The results of operations are displayed to the user.
6. **Note:**
   * The code represents binary numbers as linked lists, making it easy to perform operations like one's complement, two's complement, and addition.

Top of Form

Algorithm:

1. \*Node Class:\*

- Define a class Node with three data members: bit to store the binary bit (0 or 1), next\_node to point to the next node in the list, and prev\_node to point to the previous node.

- Initialize next\_node and prev\_node to NULL in the constructor.

2. \*List Class:\*

- Define a class List with private data members start and last to represent the doubly linked list.

- Initialize start and last to NULL in the constructor.

3. \*Input Function:\*

- Implement a function input to take a binary string as input and create a doubly linked list representing the binary number.

- Iterate through each character in the input string, create a new node, set the bit based on the character, and link it to the existing nodes.

4. \*One's Complement Function:\*

- Implement a function one\_complement that negates each bit in the binary number (1 becomes 0, and 0 becomes 1).

5. \*Two's Complement Function:\*

- Implement a function two\_complement that first calculates the one's complement and then adds 1 to the binary number.

6. \*Print Function:\*

- Implement a function print to display the binary number represented by the linked list.

7. \*Add Bit at Beginning Function:\*

- Implement a function add\_bit\_at\_begin to add a binary bit at the beginning of the linked list.

8. \*Overloaded + Operator Function:\*

- Implement an overloaded + operator function to add two binary numbers using binary addition with carry.

9. \*Main Function:\*

- Create instances of the List class (l, l1, l2).

- Use a loop to repeatedly perform operations based on user input.

- Display a menu with options for finding the one's complement, two's complement, or adding two binary numbers.

- After each operation, prompt the user if they want to try again.

- Terminate the loop when the user chooses not to continue.