**Experiment 9**

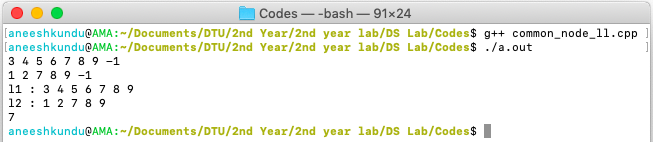
**AIM:** Write a program to find common node of 2 linked lists.

**THEORY:** A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. If node to be deleted is root, simply delete it. To delete a middle node, we must have pointer to the node previous to the node to be deleted. So, if position is not zero, we run a loop position-1 times and get pointer to the previous node.

**Code:**

1. #include < iostream >
2. #define MAX 10000
3. using namespace std;
4. struct Node {
5. int data;
6. Node \* next;
7. Node() {
8. data = 0;
9. next = NULL;
10. }
11. Node(int d) {
12. data = d;
13. next = NULL;
14. }
15. };
16. struct LinkedList {
17. Node \* head;
18. Node \* tail;
19. int len;
20. LinkedList() {
21. head = tail = NULL;
22. len = 0;
23. }
24. void insert\_at\_tail(int v) {
25. if (head == NULL) {
26. head = new Node(v);
27. tail = head;
28. return;
29. }
30. tail - > next = new Node(v);
31. tail = tail - > next;
32. len++;
33. }
34. void print() {
35. Node \* p = head;
36. if (p == NULL) return;
37. while (p != NULL) {
38. cout << p - > data << " ";
39. p = p - > next;
40. }
41. cout << endl;
42. }
43. void take\_input() {
44. int v;
45. cin >> v;
46. while (v != -1) {
47. insert\_at\_tail(v);
48. cin >> v;
49. }
50. }
51. };
52. int get\_common\_node(LinkedList l1, LinkedList l2) {
53. if (l1.tail - > data != l2.tail - > data) return -1;
54. int len1 = l1.len, len2 = l2.len;
55. int diff = abs(len1 - len2);
56. Node \* h1 = l1.head, \* h2 = l2.head;
57. while (diff) {
58. if (len1 > len2) h1 = h1 - > next;
59. else h2 = h2 - > next;
60. diff--;
61. }
62. while (h1 - > data != h2 - > data) {
63. h1 = h1 - > next;
64. h2 = h2 - > next;
65. }
66. return h1 - > data;
67. }
68. int main() {
69. LinkedList l1, l2;
70. l1.take\_input();
71. l2.take\_input();
72. cout << "l1 : ";
73. l1.print();
74. cout << "l2 : ";
75. l2.print();
76. cout << get\_common\_node(l1, l2) << endl;
77. return 0;
78. }

**Output:**

****