



Experiment-3

Student Name: Dushyant

Branch: BE-IT

Semester: 6th

Subject Name: AP Lab II

UID: 22BET10060

Section/Group: 22BET-IOT-702-B

Date of Performance: 30/01/2025

Subject Code: 22ITT-314

Problem-1

- 1. Aim:** Detect a cycle in a linked list.
- 2. Objective:** Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to.

3. Implementation/Code:

```
class Solution {
public:
    bool hasCycle(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

        while (fast != nullptr && fast->next != nullptr) {
            slow = slow->next;
            fast = fast->next->next;
            if (slow == fast)
                return true;
        }

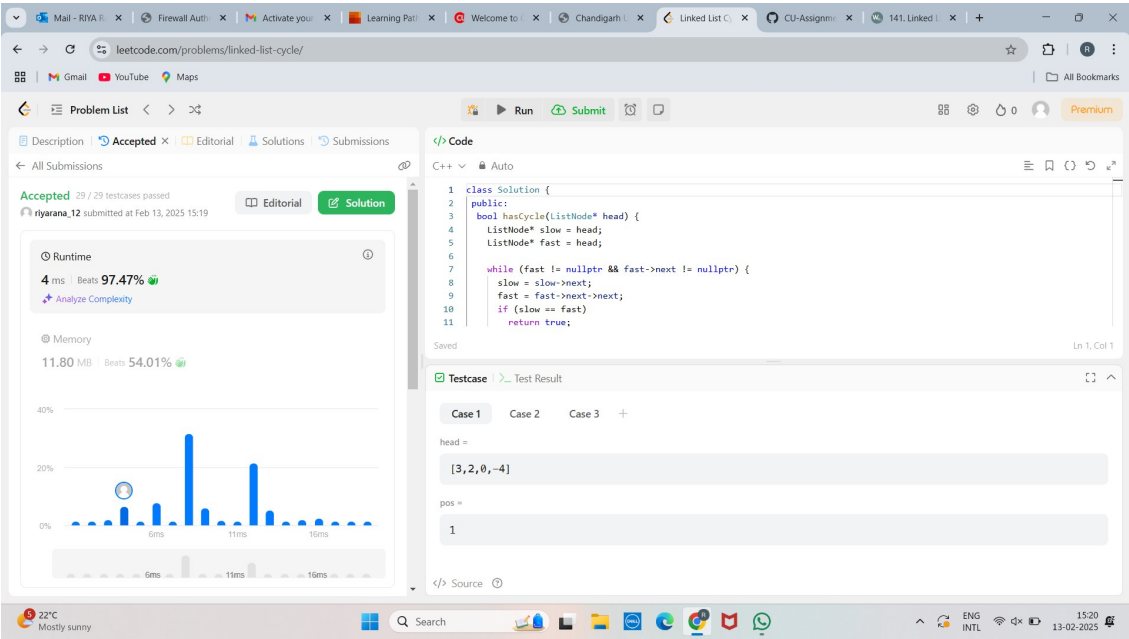
        return false;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

4. Output:



Problem-2

1. **Aim:** Reverse linked list 2
2. **Objective:** Given the head of a singly linked list and two integers left and right where $\text{left} \leq \text{right}$, reverse the nodes of the list from position left to position right, and return the reversed list.
3. **Implementation/Code:**

```
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (left == 1)
            return reverseN(head, right);

        head->next = reverseBetween(head->next, left - 1, right - 1);

        return head;
    }

private:
    ListNode* reverseN(ListNode* head, int n) {
        if (n == 1)
            return head;

        ListNode* newHead = reverseN(head->next, n - 1);
        ListNode* headNext = head->next;
        head->next = headNext->next;
        headNext->next = head;

        return newHead;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

4. Output:

The screenshot displays a web browser window with multiple tabs open, including Mail, Firewall, Activate, Learning, Welcome, Chandigarh University, Linked List, Riya121, 141, Link, Reverse, and 92. The active tab is the LeetCode problem page for 'Reverse Linked List II' (problem 92). The page shows the submission details for a user named 'riyana_12' who submitted the solution on Feb 13, 2025, at 15:37. The submission is marked as 'Accepted' with 44/44 testcases passed. The runtime is 0 ms (Beats 100.00%) and the memory usage is 11.21 MB (Beats 38.74%). A bar chart shows the performance comparison with other solutions. The input is a linked list with head = [1, 2, 3, 4, 5], left = 2, and right = 4. The output is [1, 4, 3, 2, 5]. The expected output is also shown as [1, 4, 3, 2, 5]. The bottom of the screen shows the Windows taskbar with the search bar and various application icons, including the Start menu, Search, File Explorer, Microsoft Edge, and several other apps. The system tray shows the temperature as 22°C, mostly sunny, and the time as 15:38 on 13-02-2025.

Problem-3

1. **Aim:** Rotate a list
2. **Objective:** Given the head of a linked list, rotate the list to the right by k places.
3. **Code:**

```
class Solution {  
  
    public:  
  
    ListNode* rotateRight(ListNode* head, int k) {  
        if (!head || !head->next || k == 0)  
            return head;  
  
        ListNode* tail;  
        int length = 1;  
  
        for (tail = head; tail->next; tail = tail->next)  
            ++length;  
        tail->next = head; // Circle the list.  
  
        const int t = length - k % length;  
        for (int i = 0; i < t; ++i)  
            tail = tail->next;  
        ListNode* newHead = tail->next;  
        tail->next = nullptr;  
  
        return newHead;  
    }  
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

4. Output:

The screenshot displays a web browser window showing a LeetCode submission for the 'rotate-list' problem. The submission is accepted, with a runtime of 0 ms and memory usage of 16.40 MB. The input is head = [1,2,3,4,5] and k = 2, resulting in the output [4,5,1,2,3].

Runtime: 0 ms | Beats 100.00%
Memory: 16.40 MB | Beats 65.00%

Testcase: Accepted Runtime: 0 ms

Case 1:

Input: head = [1,2,3,4,5], k = 2

Output: [4,5,1,2,3]

Expected: [4,5,1,2,3]

Problem-4

1. **Aim:** Merge k sorted lists
2. **Objective:** You are given an array of k linked-lists lists, each linked-list is sorted in ascending order. Merge all the linked-lists into one sorted linked-list and return it.

3. Implementation/Code:

```
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        ListNode dummy(0);
        ListNode* curr = &dummy;
        auto compare = [](ListNode* a, ListNode* b) { return a->val > b->val; };
        priority_queue<ListNode*, vector<ListNode*>, decltype(compare)> minHeap(
            compare);

        for (ListNode* list : lists)
            if (list != nullptr)
                minHeap.push(list);

        while (!minHeap.empty()) {
            ListNode* minNode = minHeap.top();
            minHeap.pop();
            if (minNode->next)
                minHeap.push(minNode->next);
            curr->next = minNode;
            curr = curr->next;
        }

        return dummy.next;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

4. Output:

The screenshot displays a LeetCode submission for the problem "Merge k Sorted Lists". The submission is marked as "Accepted" with 134/134 test cases passed. The user "riyarana_12" submitted it on Feb 13, 2025, at 15:52. The performance metrics show a runtime of 0 ms (Beats 100.00%) and memory usage of 18.56 MB (Beats 50.48%).

The code is written in C++ and uses a min-heap to merge the lists. The test case input is `lists = [[1,4,5], [1,3,4], [2,6]]` and the output is `[1,1,2,3,4,4,5,6]`.

```
C++  
ListNode* mergeKLists(vector<ListNode*>& lists) {  
    ListNode* dummy(0);  
    ListNode* curr = &dummy;  
    auto compare = [](ListNode* a, ListNode* b) { return a->val > b->val; };  
    priority_queue<ListNode*, vector<ListNode*&, decltype(compare)> minHeap(  
        compare);  
    while (!minHeap.empty()) {  
        ListNode* node = minHeap.top();  
        minHeap.pop();  
        curr->next = node;  
        curr = curr->next;  
        if (node->next != nullptr) minHeap.push(node->next);  
    }  
    return dummy;  
}
```


Problem-5

1. Aim: Sort List

2. Objective: Given the head of a linked list, return the list after sorting it in ascending order.

3. Implementation/Code:

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        const int length = getLength(head);
        ListNode dummy(0, head);

        for (int k = 1; k < length; k *= 2) {
            ListNode* curr = dummy.next;
            ListNode* tail = &dummy;
            while (curr != nullptr) {
                ListNode* l = curr;
                ListNode* r = split(l, k);
                curr = split(r, k);
                auto [mergedHead, mergedTail] = merge(l, r);
                tail->next = mergedHead;
                tail = mergedTail;
            }
        }

        return dummy.next;
    }
private:
    int getLength(ListNode* head) {
        int length = 0;
        for (ListNode* curr = head; curr; curr = curr->next)
            ++length;
        return length;
    }
}
```

```
ListNode* split(ListNode* head, int k) {
    while (--k && head)
        head = head->next;

    ListNode* rest = head ? head->next : nullptr;
    if (head != nullptr)
        head->next = nullptr;
    return rest;
}

pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {
    ListNode dummy(0);
    ListNode* tail = &dummy;

    while (l1 && l2) {
        if (l1->val > l2->val)
            swap(l1, l2);
        tail->next = l1;
        l1 = l1->next;
        tail = tail->next;
    }

    tail->next = l1 ? l1 : l2;
    while (tail->next != nullptr)
        tail = tail->next;

    return {dummy.next, tail};
};
```

Screenshot of the LeetCode interface showing the solution for the "Merge Two Sorted Lists" problem. The browser address bar shows the URL: `leetcode.com/problems/sort-list/submissions/1541623780/`.

The problem description on the left indicates the solution is "Accepted" for 30/30 testcases, with a runtime of 12 ms (Beats 78.71%) and memory usage of 56.91 MB (Beats 90.52%). The user "riyarana_12" submitted the solution on Feb 13, 2025, at 16:04.

The C++ code for the solution is displayed in the center:

```
C++  
// Recursion Test  
38 }  
39  
40 pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {  
41     ListNode* dummy(0);  
42     ListNode* tail = &dummy;  
43 }
```

The "Test Result" section on the right shows the solution is "Accepted" with a runtime of 0 ms. The input is `[4,2,1,3]` and the output is `[1,2,3,4]`, matching the expected result.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.