Experiment-3

Student Name: Soyash **UID: 22BET10051**
Branch: BE-IT **Section/Group: 22BET-IOT-702-B**
Semester: 6th **Date of Performance: 30/01/2025**
Subject Name: AP Lab II **Subject Code: 22ITT-314**

Problem-1

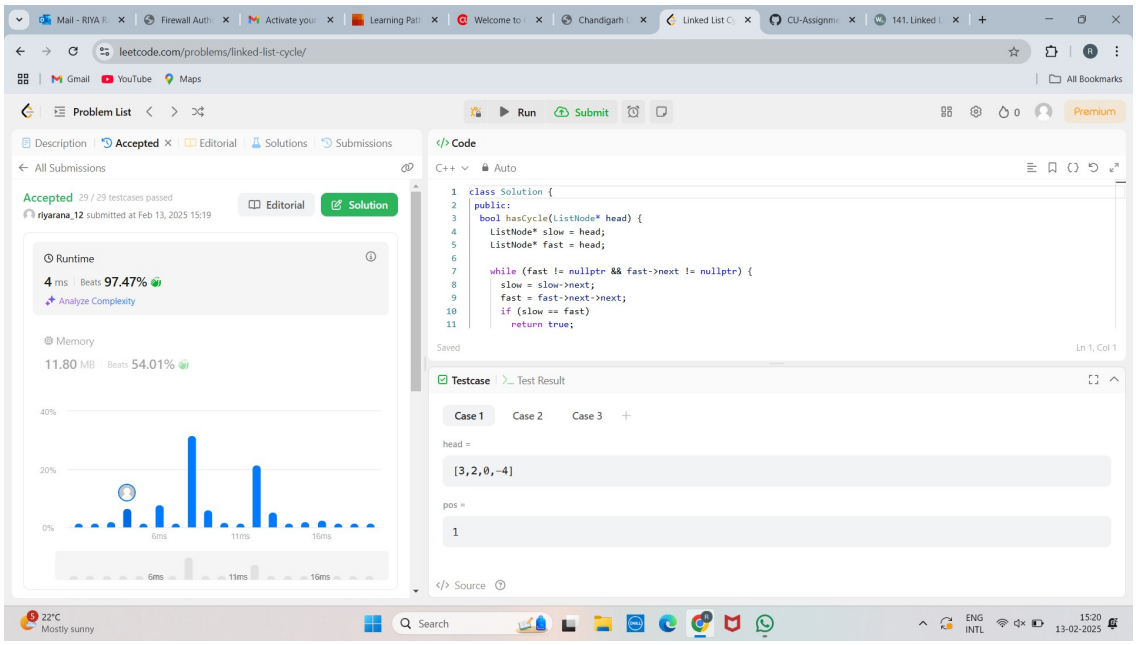
1. **Aim:** Detect a cycle in a linked list.

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;
  }
};
```

4. Output:

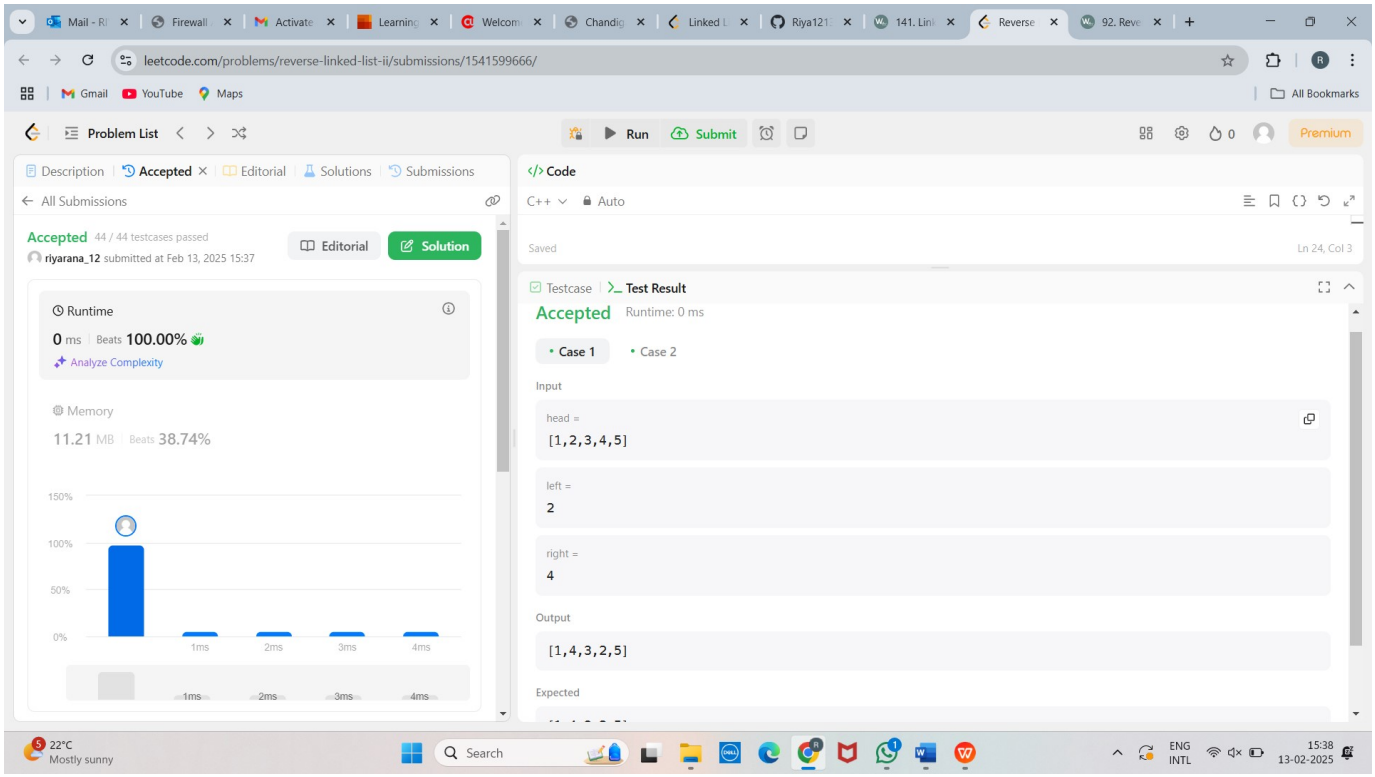

leetcode.com/problems/linked-list-cycle/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 29 / 29 testcases passed
riyarana_12 submitted at Feb 13, 2025 15:19
Editorial
Solution
Runtime
4 ms | Beats 97.47%
Analyze Complexity
Memory
11.80 MB | Beats 54.01%
Code
C++
Auto
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;
Saved
Ln 1, Col 1
Testcase
Test Result
Case 1
Case 2
Case 3
head =
[3,2,0,-4]
pos =
1
Source
Search

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 left <= 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;
  }
};
```

4. Output:


leetcode.com/problems/reverse-linked-list-ii/submissions/1541599666/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 44 / 44 testcases passed
riyarana_12 submitted at Feb 13, 2025 15:37
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
11.21 MB | Beats 38.74%
Code
C++
Auto
Saved
Ln 24, Col 3
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,4,5]
left =
2
right =
4
Output
[1,4,3,2,5]
Expected

Problem-3

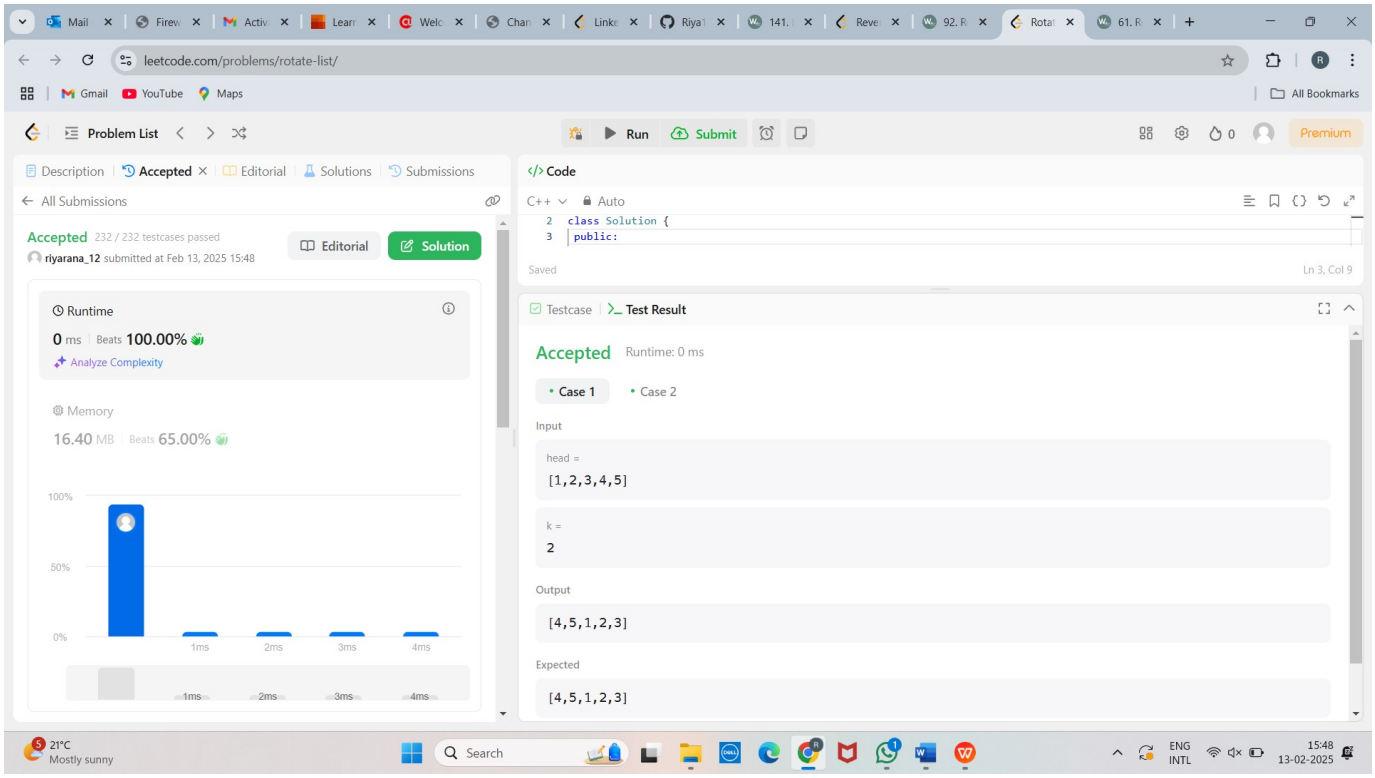
1. **Aim:** Rotate a list

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;
  }
};
```

4. Output:


leetcode.com/problems/rotate-list/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 232 / 232 testcases passed
riyarana_12 submitted at Feb 13, 2025 15:48
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
16.40 MB | Beats 65.00%
Code
C++
Auto
class Solution {
public:
Saved
Ln 3, Col 9
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,4,5]
k =
2
Output
[4,5,1,2,3]
Expected
[4,5,1,2,3]
Search
ENG
INTL
15:48
13-02-2025

Problem-4

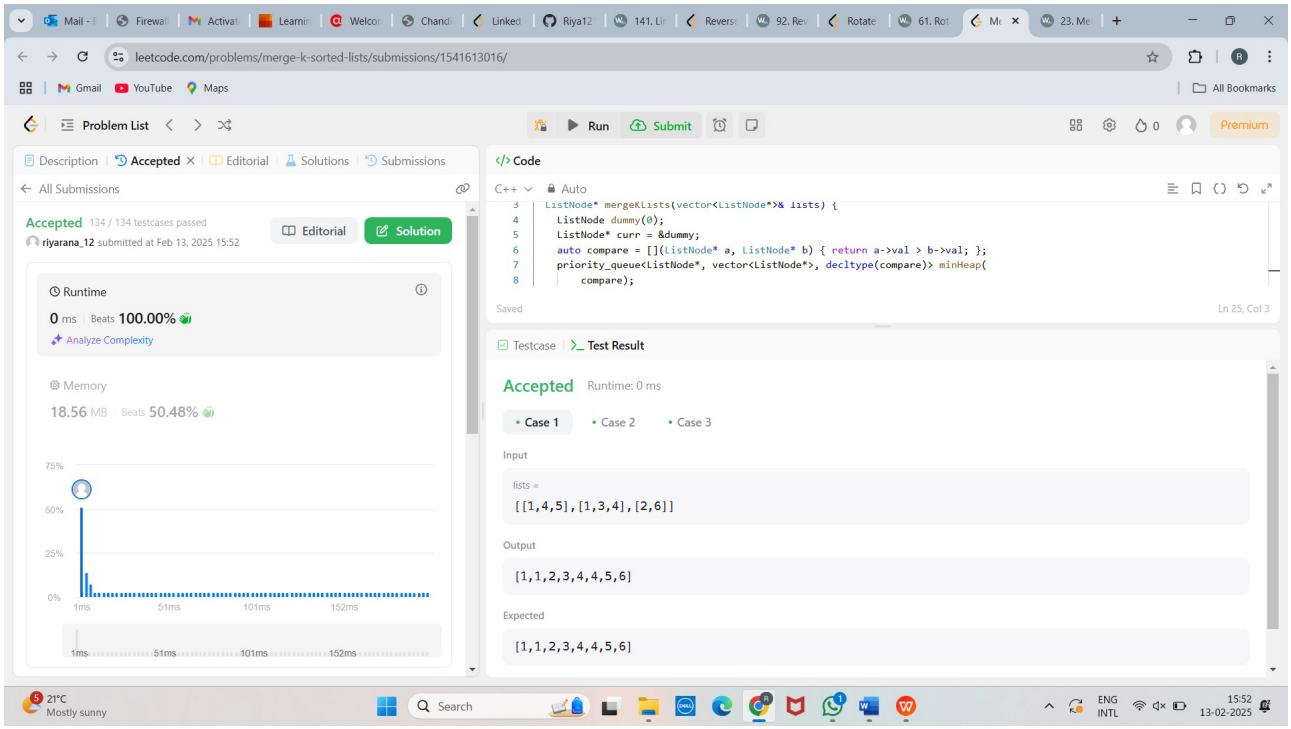
1. **Aim:** Merge k sorted lists

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;
  }
};
```

4. Output:


leetcode.com/problems/merge-k-sorted-lists/submissions/1541613016/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 134 / 134 testcases passed
riyarana_12 submitted at Feb 13, 2025 15:52
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
18.56 MB | Beats 50.48%
Code
C++
Auto
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);
Saved
Ln 25, Col 3
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
lists =
[[1,4,5],[1,3,4],[2,6]]
Output
[1,1,2,3,4,4,5,6]
Expected
[1,1,2,3,4,4,5,6]
Search
ENG
INTL
15:52
13-02-2025

Problem-5

1. Aim: Sort List

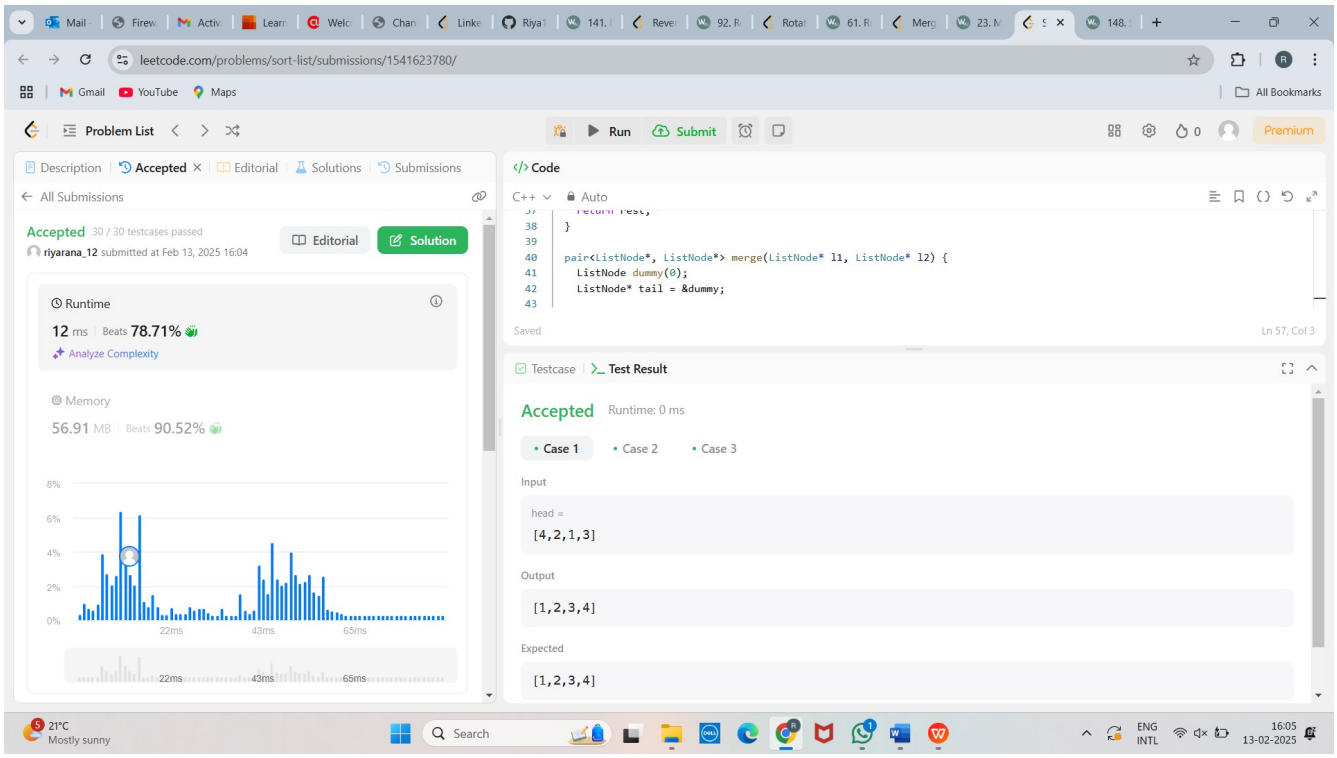
2. Objective: Given the head of a linked list, return the list after sorting it in ascending order.

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};

 }
};
```

4. Output:


leetcode.com/problems/sort-list/submissions/1541623780/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 30 / 30 testcases passed
riyarana_12 submitted at Feb 13, 2025 16:04
Editorial
Solution
Runtime
12 ms | Beats 78.71%
Analyze Complexity
Memory
56.91 MB | Beats 90.52%
Code
C++
Auto
pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {
ListNode dummy(0);
ListNode* tail = &dummy;
Saved
Ln 57, Col 3
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[4,2,1,3]
Output
[1,2,3,4]
Expected
[1,2,3,4]
Search
ENG
INTL
16:05
13-02-2025