NAME: MOAZZAM FAROOQUI

ROLL NO: CT-24068

COURSE CODE: CT-159

ASSIGNMENT: DSA PRACTICE SESSION TASK#1

Q1. Find first and last position of elements in a sorted array.

SOURCE CODE:

```cpp
class Solution {
public:
    int firstOccurrence(vector<int>& nums,int target){
        int low=0;
        int high=nums.size()-1;
        int first=-1;

        while (low<=high){
            int mid=(low+high)/2;

            if (nums[mid]==target){
                first=mid;
                high=mid-1;
            }
            else if(nums[mid]<target){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return first;
    }
```

```cpp
    int lastOccurrence(vector<int>& nums,int target){
        int low=0;
        int high=nums.size()-1;
        int last=-1;

        while (low <= high){
            int mid=(low+high)/2;

            if (nums[mid]==target){
                last=mid;
                low=mid+1;
            }
            else if(nums[mid]<target){
                low=mid+1;
            }
            else {
                high=mid-1;
            }
        }
        return last;
    }
```

```
    vector<int>searchRange(vector<int>&nums,int target){
        int first=firstOccurrence(nums,target);
        int last=lastOccurrence(nums,target);
        return{first,last};
    }
};
```

ACCEPTANCE STATUS:

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =
[5,7,7,8,8,10]

target =
8

Output

[3,4]

Expected

[3,4]

Contribute a testcase

Q2. Search a 2D Matrix

SOURCE CODE:

Code

C++ Auto

```cpp
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int m=matrix.size();
        int n=matrix[0].size();
        int low=0;
        int high=m*n-1;
        while(low<=high){
            int mid=(low+high)/2;
            int row=mid/n;
            int col=mid%n;
            int value=matrix[row][col];
            if(value==target){
                return true;
            }
            else if(value<target){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return false;
    }
};
```

ACCEPTANCE STATUS:

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

matrix =
[[1,3,5,7],[10,11,16,20],[23,30,34,60]]

target =
3

Output

true

Expected

true

Contribute a testcase

Q3. Koko Eating Bananas

SOURCE CODE:

C++ Auto

```cpp
class Solution{
public:
    int minEatingSpeed(vector<int>&piles,int h){
        int low=1;
        int high=*max_element(piles.begin(),piles.end());
        int ans=high;

        while (low<=high){
            int mid=(low+high)/2;
            int currenthours=0;

            for (int &p:piles) {
                currenthours+=(p+mid-1)/mid;
            }
            if(currenthours>h) {
                low=mid+1;
            }
            else {
                ans=mid;
                high=mid-1;
            }
        }
        return ans;
    }
```

ACCEPTANCE STATUS:

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

piles =
[3,6,7,11]

h =
8

Output

4

Expected

4

Contribute a testcase

Q4. Merge Two Sorted Lists

SOURCE CODE:

C++ Auto

```cpp
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* head1, ListNode* head2) {
        if(head1==nullptr){
            return head2;
        }
        if(head2==nullptr){
            return head1;
        }
        if(head1->val<=head2->val){
            head1->next=mergeTwoLists(head1->next,head2);
            return head1;
        }
        else{
            head2->next=mergeTwoLists(head1,head2->next);
            return head2;
        }
    }
};

```

ACCEPTANCE STATUS:

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

list1 =

[1,2,4]

list2 =

[1,3,4]

Output

[1,1,2,3,4,4]

Expected

[1,1,2,3,4,4]

Contribute a testcase

Q5. Remove Nth Node From End of List

SOURCE CODE:

Code

C++ Auto

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head,int n) {
        ListNode*dummy=new ListNode(0);
        dummy->next=head;

        ListNode*fast=dummy;
        ListNode*slow=dummy;

        for(int i=0;i<=n;i++){
            fast=fast->next;
        }
```

```cpp
        while(fast!=nullptr){
            fast=fast->next;
            slow=slow->next;
        }
        slow->next=slow->next->next;
        return dummy->next;
    }
};
```

ACCEPTANCE STATUS:

Testcase | >_ Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

head =
[1,2]

n =
1

Output

[1]

Expected

[1]

Contribute a testcase

6. Linked List Cycle II

SOURCE CODE:

Code

C++ | Auto

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        if(head==nullptr || head->next==nullptr){
            return nullptr;
        }
        ListNode*slow=head;
        ListNode*fast=head;

        while(fast!=nullptr && fast->next!=nullptr){
            slow=slow->next;
            fast=fast->next->next;
```

```
        if(slow==fast){
            ListNode*entry=head;
            while(entry!=slow){
                entry=entry->next;
                slow=slow->next;
            }
            return entry;
        }
    }
    return nullptr;
}
};
```

ACCEPTANCE STATUS:

Testcase | Test Result

Accepted Runtime: 2 ms

• Case 1 • Case 2 • Case 3

Input

head =

[3,2,0,-4]

pos =

1

Output

tail connects to node index 1

Expected

tail connects to node index 1

Contribute a testcase