



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Assignment 2

Student Name: Aryan Mehta

Branch: BE-CSE (General)

Semester: 6th

Subject Name: Advanced Programming Lab-2

UID: 22BCS12209

Section/Group: FL_IOT-602 A

Date of Performance: 04-02-25

Subject Code: 22CSP-351

1. **Aim:** 1763. Longest Nice Substring

Implementation/ Code:

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if(s.size()<2) return "";
        unordered_set<char> charSet(s.begin(), s.end());
        for(int i=0; i<s.size(); i++){
            if(charSet.count(tolower(s[i]))&&charSet.count(toupper(s[i]))){
                continue;
            }
            string left=longestNiceSubstring(s.substr(0, i));
            string right=longestNiceSubstring(s.substr(i+1));
            return left.size() >= right.size() ? left:right;
        }
        return s;
    }
};
```

Output:

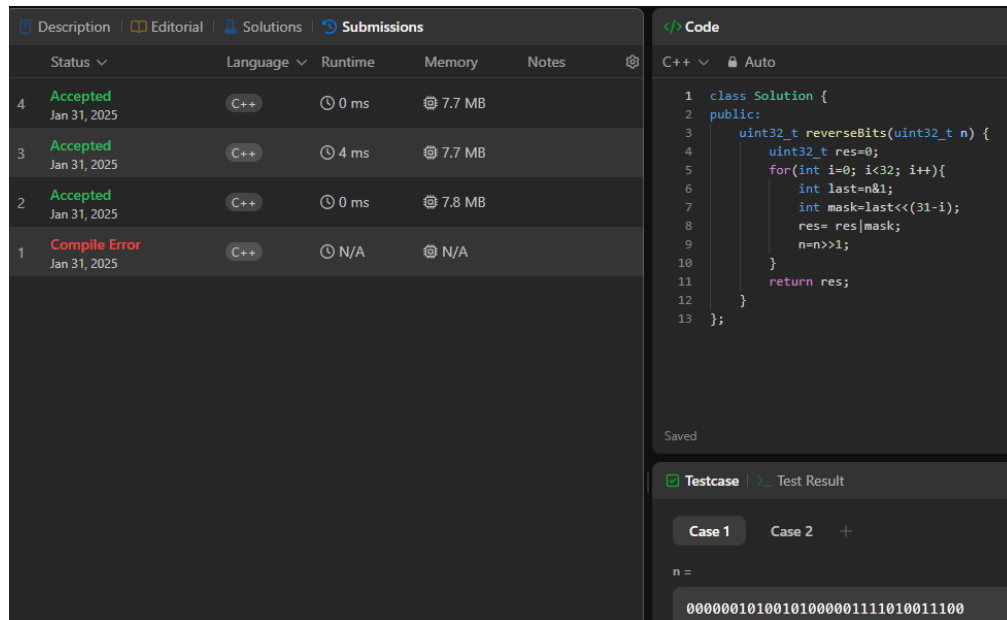
The screenshot shows a C++ IDE with a dark theme. The left pane displays the code for the 'longestNiceSubstring' function, which is identical to the one provided in the text. The right pane shows the execution output for a test case where the input string 's' is 'YazaAay'. The output is 'YazaAay', indicating that the function correctly identifies the longest nice substring.

2. Aim: 190. Reverse Bits

Implementation/ Code:

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t res=0;
        for(int i=0; i<32; i++){
            int last=n&1;
            int mask=last<<(31-i);
            res= res|mask;
            n=n>>1;
        }
        return res;
    }
};
```

Output:



The screenshot shows a coding platform interface. On the left, the 'Submissions' tab is active, displaying a table of submission results:

	Status	Language	Runtime	Memory	Notes
4	Accepted Jan 31, 2025	C++	0 ms	7.7 MB	
3	Accepted Jan 31, 2025	C++	4 ms	7.7 MB	
2	Accepted Jan 31, 2025	C++	0 ms	7.8 MB	
1	Compile Error Jan 31, 2025	C++	N/A	N/A	

On the right, the 'Code' tab is active, showing the C++ code for the solution:

```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         uint32_t res=0;
5         for(int i=0; i<32; i++){
6             int last=n&1;
7             int mask=last<<(31-i);
8             res= res|mask;
9             n=n>>1;
10        }
11        return res;
12    }
13};
```

Below the code, the 'Testcase' tab is active, showing the test result for Case 1:

Case 1 Case 2 +

n =

00000010100101000001111010011100

3. Aim: 191. Number of 1 Bits

Implementation/ Code:

```
class Solution {
public:
    int hammingWeight(int n) {
        int count=0;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
while(n){
    count+=(n&1);
    n>>=1;
}
return count;
}
};
```

Output:

The screenshot shows a code editor interface with a submission status of 'Accepted'. The code is in C++ and implements a function to calculate the Hamming weight of a number. The code is as follows:

```
1 class Solution {
2 public:
3     int hammingWeight(int n) {
4         int count=0;
5         while(n){
6             count+=(n&1);
7             n>>=1;
8         }
9         return count;
10    }
11 };
```

4. Aim: 53. Maximum Subarray

Implementation/ Code:

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = INT_MIN;
        int currentSum = 0;
        for (int num : nums) {
            currentSum += num;
            maxSum = max(maxSum, currentSum);
            if (currentSum < 0) currentSum = 0;
        }
        return maxSum;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

};

Output:

```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int maxSum = INT_MIN;
5         int currentSum = 0;
6
7         for (int num : nums) {
8             currentSum += num;
9             maxSum = max(maxSum, currentSum);
10            if (currentSum < 0) currentSum = 0;
11        }
12        return maxSum;
13    }
14 };
15
```

Testcase | Test Result

Case 1 Case 2 Case 3 +

nums =

[-2,1,-3,4,-1,2,1,-5,4]

5. Aim: 240. Search a 2D Matrix II

Implementation/ Code:

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int row = 0, col = matrix[0].size() - 1;
        while (row < matrix.size() && col >= 0) {
            if (matrix[row][col] == target) return true;
            matrix[row][col] > target ? col-- : row++;
        }
        return false;
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int row = 0, col = matrix[0].size() - 1;
5         while (row < matrix.size() && col >= 0) {
6             if (matrix[row][col] == target) return true;
7             if (matrix[row][col] > target) col-- : row++;
8         }
9         return false;
10    }
11 };
12
```

Testcase

Case 1 Case 2 +

matrix =

[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]

target =

5

6. Aim: 372. Super Pow

Implementation/ Code:

```
class Solution {
public:
    int modPow(int a, int b, int mod) {
        int res = 1;
        a %= mod;
        while (b) {
            if (b % 2) res = (1LL * res * a) % mod;
            a = (1LL * a * a) % mod;
            b /= 2;
        }
        return res;
    }

    int superPow(int a, vector<int>& b) {
        int mod = 1337, res = 1;
        for (int d : b) res = modPow(res, 10, mod) * modPow(a, d, mod) % mod;
        return res;
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

The screenshot shows a C++ submission interface. On the left, a table lists submissions with columns for Status, Language, Runtime, Memory, and Notes. The first submission is 'Accepted' in C++ with a runtime of 0 ms and memory of 15.3 MB. On the right, the 'Code' editor shows a C++ solution with a `modPow` function and a `superPow` function. Below the code editor, the 'Testcase' section shows 'Case 1' with input `a = 2` and `b = [3]`.

```
class Solution {
public:
    int modPow(int a, int b, int mod) {
        int res = 1;
        a %= mod;
        while (b) {
            if (b % 2) res = (1LL * res * a) % mod;
            a = (1LL * a * a) % mod;
            b /= 2;
        }
        return res;
    }

    int superPow(int a, vector<int>& b) {
        int mod = 1337, res = 1;
        for (int d : b) res = modPow(res, 10, mod) * modPow(a, d, mod) % mod;
        return res;
    }
};
```

7. Aim: 932. Beautiful Array

Implementation/ Code:

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res = {1};
        while (res.size() < n) {
            vector<int> tmp;
            for (int x : res) if (2 * x - 1 <= n) tmp.push_back(2 * x - 1);
            for (int x : res) if (2 * x <= n) tmp.push_back(2 * x);
            res = tmp;
        }
        return res;
    }
};
```

Output:



```
1 Accepted
5 minutes ago
C++
0 ms
10 MB

2 public:
3     vector<int> beautifulArray(int n) {
4         vector<int> res = {1};
5         while (res.size() < n) {
6             vector<int> tmp;
7             for (int x : res) if (2 * x - 1 <= n) tmp.push_back(2 * x - 1);
8             for (int x : res) if (2 * x <= n) tmp.push_back(2 * x);
9             res = tmp;
10        }
11        return res;
12    }
13 };
14
```

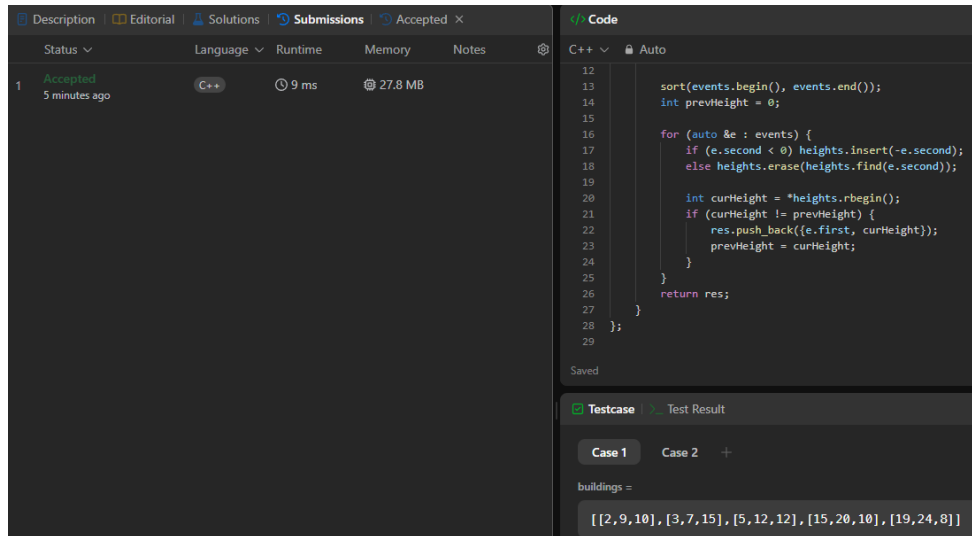
8. Aim: 218. The Skyline Problem

Implementation/ Code:

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        multiset<int> heights = {0};
        vector<vector<int>> res;
        for (auto &b : buildings) {
            events.emplace_back(b[0], -b[2]);
            events.emplace_back(b[1], b[2]);
        }
        sort(events.begin(), events.end());
        int prevHeight = 0;
        for (auto &e : events) {
            if (e.second < 0) heights.insert(-e.second);
            else heights.erase(heights.find(e.second));

            int curHeight = *heights.rbegin();
            if (curHeight != prevHeight) {
                res.push_back({e.first, curHeight});
                prevHeight = curHeight;
            }
        }
        return res;
    }
};
```

Output:



The screenshot shows a C++ IDE with a submission status of 'Accepted' 5 minutes ago. The code is as follows:

```

12
13     sort(events.begin(), events.end());
14     int prevHeight = 0;
15
16     for (auto &e : events) {
17         if (e.second < 0) heights.insert(-e.second);
18         else heights.erase(heights.find(e.second));
19
20         int curHeight = *heights.rbegin();
21         if (curHeight != prevHeight) {
22             res.push_back({e.first, curHeight});
23             prevHeight = curHeight;
24         }
25     }
26     return res;
27 }
28 };
29

```

Below the code, the 'Testcase' tab shows the input for 'Case 1':

```

buildings =
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]

```

9. Aim: 493. Reverse Pairs

Implementation/ Code:

```

class Solution {
public:
    int merge(vector<int>& nums, int left, int mid, int right) {
        int count = 0, j = mid + 1;
        for (int i = left; i <= mid; i++) {
            while (j <= right && nums[i] > 2LL * nums[j]) j++;
            count += j - (mid + 1);
        }

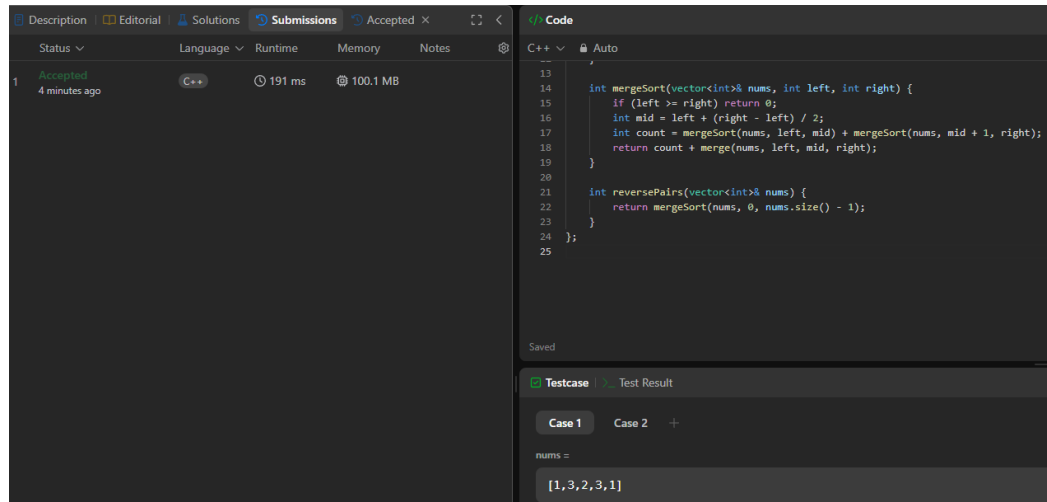
        inplace_merge(nums.begin() + left, nums.begin() + mid + 1, nums.begin() + right + 1);
        return count;
    }

    int mergeSort(vector<int>& nums, int left, int right) {
        if (left >= right) return 0;
        int mid = left + (right - left) / 2;
        int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
        return count + merge(nums, left, mid, right);
    }

    int reversePairs(vector<int>& nums) {
        return mergeSort(nums, 0, nums.size() - 1);
    }
};

```


Output:



The screenshot shows a C++ IDE with a dark theme. The left pane displays the submission status: 'Accepted' 4 minutes ago, C++, 191 ms, 100.1 MB. The right pane shows the code for a merge sort algorithm. Below the code, the 'Testcase' section is visible, showing 'Case 1' with the input array [1, 3, 2, 3, 1].

```

13
14 int mergeSort(vector<int>& nums, int left, int right) {
15     if (left >= right) return 0;
16     int mid = left + (right - left) / 2;
17     int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
18     return count + merge(nums, left, mid, right);
19 }
20
21 int reversePairs(vector<int>& nums) {
22     return mergeSort(nums, 0, nums.size() - 1);
23 }
24 };
25

```

Testcase: Case 1
nums = [1, 3, 2, 3, 1]

10. Aim: 2407. Longest Increasing Subsequence II

Implementation/ Code:

```

class SegmentTree {
public:
    vector<int> tree;
    int size;

    SegmentTree(int n) {
        size = n;
        tree.resize(4 * n, 0);
    }

    void update(int index, int value, int node = 1, int start = 0, int end = 100000) {
        if (start == end) {
            tree[node] = value;
            return;
        }
        int mid = (start + end) / 2;
        if (index <= mid) update(index, value, 2 * node, start, mid);
        else update(index, value, 2 * node + 1, mid + 1, end);
        tree[node] = max(tree[2 * node], tree[2 * node + 1]);
    }

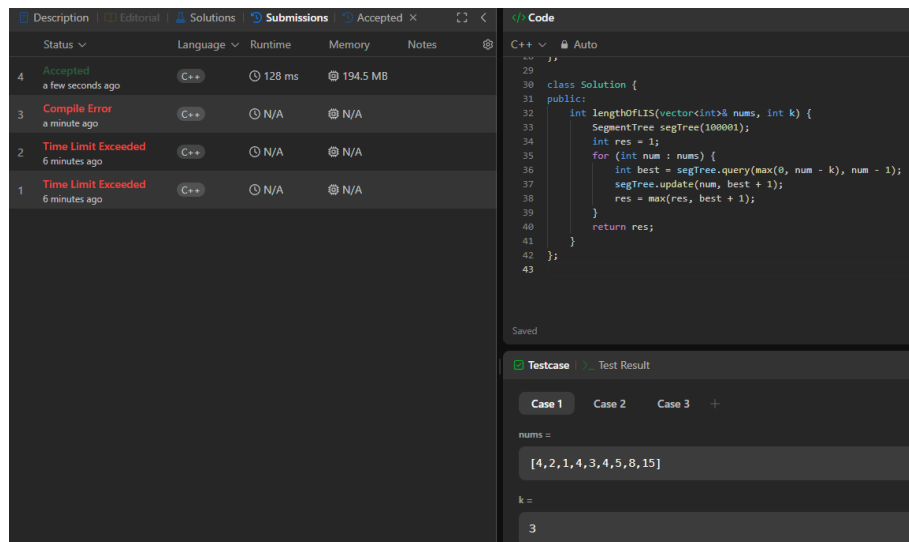
    int query(int left, int right, int node = 1, int start = 0, int end = 100000) {
        if (left > end || right < start) return 0;
        if (left <= start && end <= right) return tree[node];
        int mid = (start + end) / 2;

```

```
        return max(query(left, right, 2 * node, start, mid), query(left, right, 2 * node + 1, mid + 1,
end));
    }
};
```

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        SegmentTree segTree(100001);
        int res = 1;
        for (int num : nums) {
            int best = segTree.query(max(0, num - k), num - 1);
            segTree.update(num, best + 1);
            res = max(res, best + 1);
        }
        return res;
    }
};
```

Output:



The screenshot shows a code editor with the following code:

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        SegmentTree segTree(100001);
        int res = 1;
        for (int num : nums) {
            int best = segTree.query(max(0, num - k), num - 1);
            segTree.update(num, best + 1);
            res = max(res, best + 1);
        }
        return res;
    }
};
```

The test case shows:

```
nums = [4, 2, 1, 4, 3, 4, 5, 8, 15]
k = 3
```

The output is 4.

11. Aim: 88. Merge Sorted Array

Implementation/ Code:

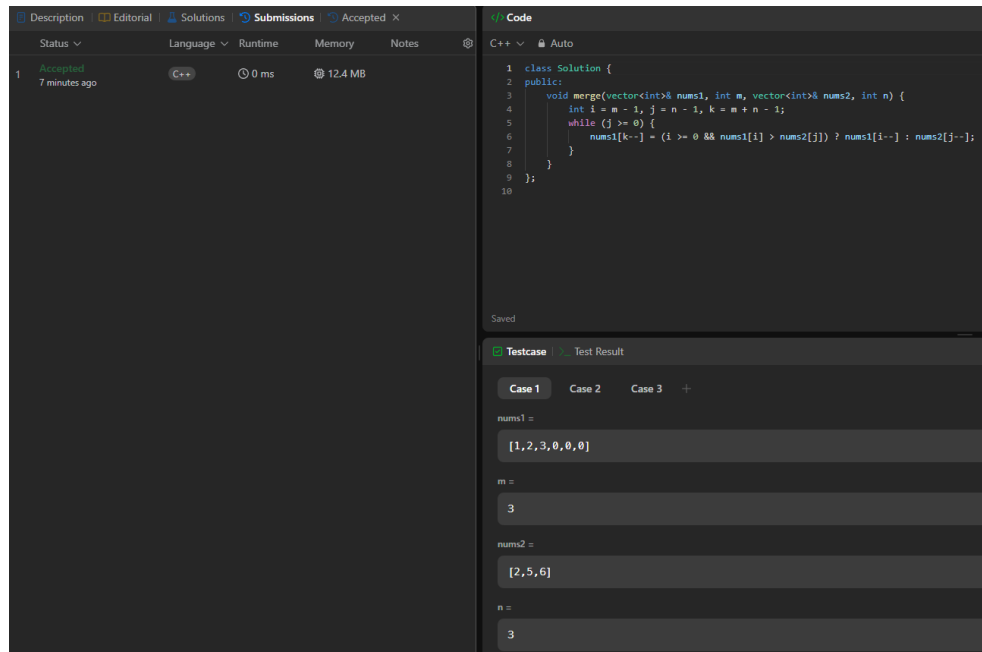
```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1, j = n - 1, k = m + n - 1;
        while (j >= 0) {
```

```

        nums1[k--] = (i >= 0 && nums1[i] > nums2[j]) ? nums1[i--] : nums2[j--];
    }
}
};

```

Output:



The screenshot shows a C++ IDE with a submission status of 'Accepted' and a runtime of 0 ms. The code implements a merge sort algorithm. The test case results show the following input and output:

```

nums1 = [1,2,3,0,0,0]
m = 3
nums2 = [2,5,6]
n = 3

```

12. Aim: 278. First Bad Version

Implementation/ Code:

```

class Solution {
public:
    int firstBadVersion(int n) {
        int left = 1, right = n;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) right = mid;
            else left = mid + 1;
        }
        return left;
    }
};

```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1 class Solution {
2 public:
3     int firstBadVersion(int n) {
4         int left = 1, right = n;
5         while (left < right) {
6             int mid = left + (right - left) / 2;
7             if (isBadVersion(mid)) right = mid;
8             else left = mid + 1;
9         }
10        return left;
11    }
12 };
13
```

Testcase

Case 1 Case 2 +

n =

5

bad =

4

13. Aim: 75. Sort Colors

Implementation/ Code:

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0, high = nums.size() - 1;
        while (mid <= high) {
            if (nums[mid] == 0) swap(nums[low++], nums[mid++]);
            else if (nums[mid] == 1) mid++;
            else swap(nums[mid], nums[high--]);
        }
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1 class Solution {
2 public:
3     void sortColors(vector<int>& nums) {
4         int low = 0, mid = 0, high = nums.size() - 1;
5         while (mid <= high) {
6             if (nums[mid] == 0) swap(nums[low++], nums[mid++]);
7             else if (nums[mid] == 1) mid++;
8             else swap(nums[mid], nums[high--]);
9         }
10    }
11 };
12
```

14. Aim: 347. Top K Frequent Elements

Implementation/ Code:

```
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        unordered_map<int, int> freq;
        for (int num : nums) freq[num]++;

        priority_queue<pair<int, int>> pq;
        for (auto [num, count] : freq) pq.emplace(count, num);

        vector<int> res;
        while (k--) {
            res.push_back(pq.top().second);
            pq.pop();
        }
        return res;
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
2 public:
3     vector<int> topKFrequent(vector<int>& nums, int k) {
4         unordered_map<int, int> freq;
5         for (int num : nums) freq[num]++;
6
7         priority_queue<pair<int, int>> pq;
8         for (auto [num, count] : freq) pq.emplace(count, num);
9
10        vector<int> res;
11        while (k-- > 0) {
12            res.push_back(pq.top().second);
13            pq.pop();
14        }
15        return res;
16    }
17 };
18
```

Testcase 1: Test Result

Case 1 Case 2 +

nums =

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

k =

2

15. Aim: 215. Kth Largest Element in an Array

Implementation/ Code:

```
class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
        priority_queue<int, vector<int>, greater<int>> pq;
        for (int num : nums) {
            pq.push(num);
            if (pq.size() > k) pq.pop();
        }
        return pq.top();
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1 class Solution {
2 public:
3     int findKthLargest(vector<int>& nums, int k) {
4         priority_queue<int, vector<int>, greater<int>> pq;
5         for (int num : nums) {
6             pq.push(num);
7             if (pq.size() > k) pq.pop();
8         }
9         return pq.top();
10    }
11 };
12
```

Testcase | Test Result

Case 1 Case 2 +

nums =
[3,2,1,5,6,4]

k =
2

16. Aim: 162. Find Peak Element

Implementation/ Code:

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int left = 0, right = nums.size() - 1;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] > nums[mid + 1]) right = mid;
            else left = mid + 1;
        }
        return left;
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1 class Solution {
2 public:
3     int findPeakElement(vector<int>& nums) {
4         int left = 0, right = nums.size() - 1;
5         while (left < right) {
6             int mid = left + (right - left) / 2;
7             if (nums[mid] > nums[mid + 1]) right = mid;
8             else left = mid + 1;
9         }
10        return left;
11    }
12 };
13
```

Testcase | Test Result

Case 1 Case 2 +

nums =

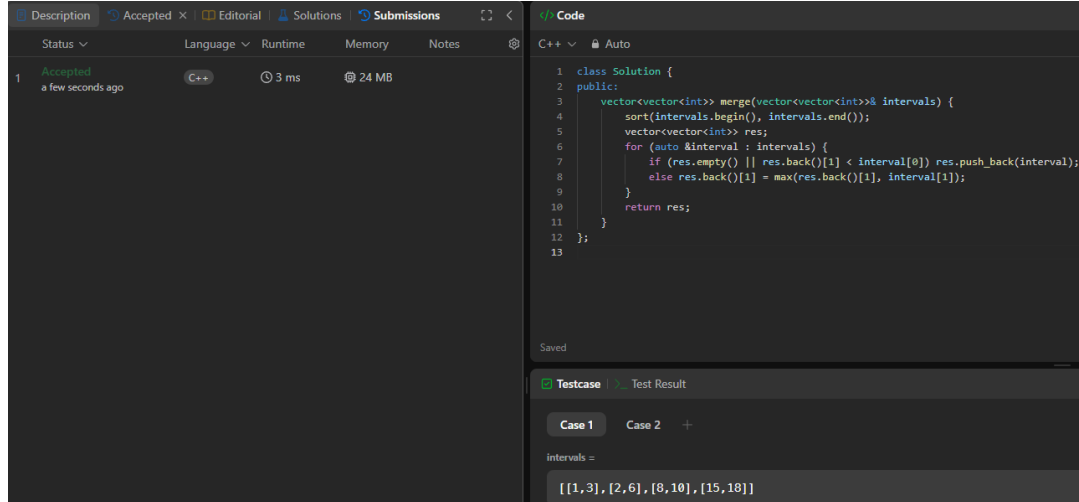
[1, 2, 3, 1]

17. Aim: 56. Merge Intervals

Implementation/ Code:

```
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& intervals) {
        sort(intervals.begin(), intervals.end());
        vector<vector<int>> res;
        for (auto &interval : intervals) {
            if (res.empty() || res.back()[1] < interval[0]) res.push_back(interval);
            else res.back()[1] = max(res.back()[1], interval[1]);
        }
        return res;
    }
};
```

Output:



```
1 class Solution {
2 public:
3     vector<vector<int>> merge(vector<vector<int>>& intervals) {
4         sort(intervals.begin(), intervals.end());
5         vector<vector<int>> res;
6         for (auto &interval : intervals) {
7             if (res.empty() || res.back()[1] < interval[0]) res.push_back(interval);
8             else res.back()[1] = max(res.back()[1], interval[1]);
9         }
10        return res;
11    }
12 };
13
```

Testcase | Test Result

Case 1 Case 2 +

intervals =

[[1,3], [2,6], [8,10], [15,18]]

18. Aim: 33.Search in Rotated Sorted Array

Implementation/ Code:

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left = 0, right = nums.size() - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) return mid;

            if (nums[left] <= nums[mid]) {
                if (nums[left] <= target && target < nums[mid]) right = mid - 1;
                else left = mid + 1;
            } else {
                if (nums[mid] < target && target <= nums[right]) left = mid + 1;
                else right = mid - 1;
            }
        }
        return -1;
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

The screenshot shows a C++ IDE with a submission status window on the left and a code editor on the right. The status window shows a submission is 'Accepted' with a runtime of 0 ms and memory of 15.1 MB. The code editor shows a binary search function. Below the code, the 'Testcase' section is visible, showing an input array [4, 5, 6, 7, 0, 1, 2] and a target value of 0.

```
1 class Solution {
2 public:
3     int search(vector<int>& nums, int target) {
4         int left = 0, right = nums.size() - 1;
5         while (left <= right) {
6             int mid = left + (right - left) / 2;
7             if (nums[mid] == target) return mid;
8
9             if (nums[left] <= nums[mid]) {
10                 if (nums[left] <= target && target < nums[mid]) right = mid - 1;
11                 else left = mid + 1;
12             } else {
13                 if (nums[mid] < target && target <= nums[right]) left = mid + 1;
14                 else right = mid - 1;
15             }
16         }
17         return -1;
18     }
19 }
```

Testcase

Case 1 Case 2 Case 3 +

nums =

[4, 5, 6, 7, 0, 1, 2]

target =

0

19. Aim: 324. Wiggle Sort II

Implementation/ Code:

```
class Solution {
public:
    void wiggleSort(vector<int>& nums) {
        vector<int> sorted = nums;
        sort(sorted.begin(), sorted.end());
        int n = nums.size(), mid = (n - 1) / 2, end = n - 1;

        for (int i = 0; i < n; i++) {
            nums[i] = (i % 2 == 0) ? sorted[mid--] : sorted[end--];
        }
    }
};
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

The screenshot shows a C++ IDE with a submission status of 'Accepted' and a runtime of 1 ms. The code implements a wiggle sort algorithm. The test case input is [1, 5, 1, 1, 6, 4].

```
1 class Solution {
2 public:
3     void wiggleSort(vector<int>& nums) {
4         vector<int> sorted = nums;
5         sort(sorted.begin(), sorted.end());
6         int n = nums.size(), mid = (n - 1) / 2, end = n - 1;
7
8         for (int i = 0; i < n; i++) {
9             nums[i] = (i % 2 == 0) ? sorted[mid--] : sorted[end--];
10        }
11    }
12 };
13
```

Testcase: Case 1, Case 2, +

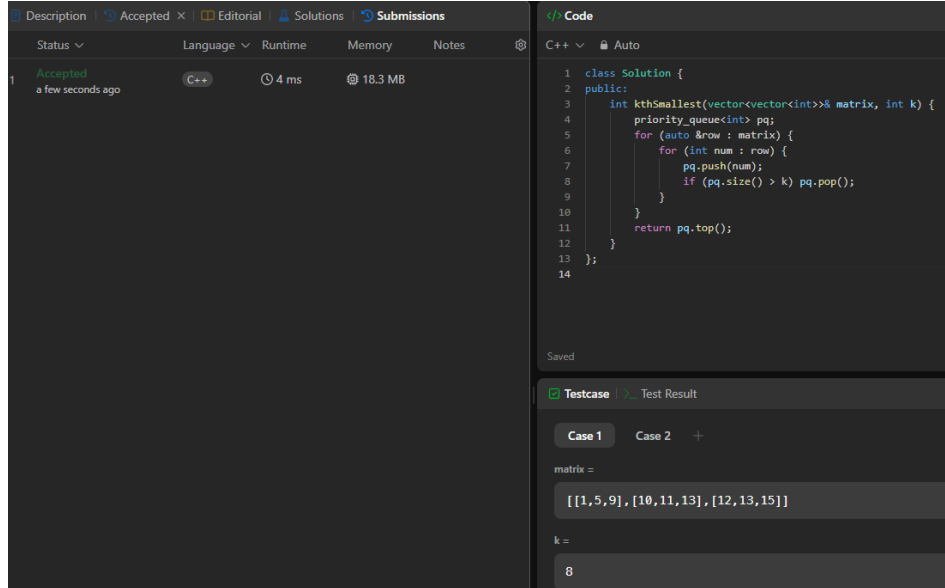
nums = [1, 5, 1, 1, 6, 4]

20. Aim: 378.Kth Smallest Element in a Sorted Matrix

Implementation/ Code:

```
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k) {
        priority_queue<int> pq;
        for (auto &row : matrix) {
            for (int num : row) {
                pq.push(num);
                if (pq.size() > k) pq.pop();
            }
        }
        return pq.top();
    }
};
```

Output:



```

1 class Solution {
2 public:
3     int kthSmallest(vector<vector<int>>& matrix, int k) {
4         priority_queue<int> pq;
5         for (auto &row : matrix) {
6             for (int num : row) {
7                 pq.push(num);
8                 if (pq.size() > k) pq.pop();
9             }
10        }
11        return pq.top();
12    }
13 };
14

```

Testcase / Test Result

Case 1 Case 2 +

matrix =

[[1,5,9],[10,11,13],[12,13,15]]

k =

8

21. Aim: 4. Median of Two Sorted Arrays

Implementation/ Code:

```

class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        if (nums1.size() > nums2.size()) return findMedianSortedArrays(nums2, nums1);

        int x = nums1.size(), y = nums2.size();
        int low = 0, high = x;

        while (low <= high) {
            int partitionX = (low + high) / 2;
            int partitionY = (x + y + 1) / 2 - partitionX;

            int maxX = (partitionX == 0) ? INT_MIN : nums1[partitionX - 1];
            int minX = (partitionX == x) ? INT_MAX : nums1[partitionX];
            int maxY = (partitionY == 0) ? INT_MIN : nums2[partitionY - 1];
            int minY = (partitionY == y) ? INT_MAX : nums2[partitionY];

            if (maxX <= minY && maxY <= minX) {
                if ((x + y) % 2 == 0) return (max(maxX, maxY) + min(minX, minY)) / 2.0;
                else return max(maxX, maxY);
            } else if (maxX > minY) {
                high = partitionX - 1;
            } else {
                low = partitionX + 1;
            }
        }
    }
}

```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }  
    return -1;  
}  
};
```

Output:

The screenshot shows a C++ IDE interface. On the left, a 'Submissions' tab is active, showing a submission with status 'Accepted', language 'C++', runtime '0 ms', and memory '94.9 MB'. On the right, the 'Code' editor displays the following C++ code:

```
1 class Solution {  
2 public:  
3     double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {  
4         if (nums1.size() > nums2.size()) return findMedianSortedArrays(nums2, nums1);  
5  
6         int x = nums1.size(), y = nums2.size();  
7         int low = 0, high = x;  
8  
9         while (low <= high) {  
10            int partitionX = (low + high) / 2;  
11            int partitionY = (x + y + 1) / 2 - partitionX;  
12  
13            int maxX = (partitionX == 0) ? INT_MIN : nums1[partitionX - 1];  
14            int minX = (partitionX == x) ? INT_MAX : nums1[partitionX];  
15            int maxY = (partitionY == 0) ? INT_MIN : nums2[partitionY - 1];  
16            int minY = (partitionY == y) ? INT_MAX : nums2[partitionY];  
17  
18            ...  
19        }  
20    }  
21 };
```

Below the code editor, the 'Testcase' tab is active, showing two test cases:

Case 1: nums1 = [1,3], nums2 = [2]

Case 2: (empty)