



DOMAIN WINTER CAMP WORKSHEET

DAY-2

Student Name: Riya Kungotra

Branch: BE-CSE

Semester: 5th

UID: 22BCS14298

Section/Group: FL_IOT_603 (B)

Date: 20/12/2024

Very Easy

Q 1 : Majority Elements

Given an array nums of size n, return the majority element.

The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ times. You may assume that the majority element always exists in the array.

SOLUTION :-

```
#include <vector>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int majorityElement(vector<int>& nums) {
```

```
    int candidate = 0, count = 0;
```

```
    // Phase 1: Find the candidate for majority element
```

```
    for (int num : nums) {
```

```
        if (count == 0) {
```

```
            candidate = num;
```

```
        }
```

```
        count += (num == candidate) ? 1 : -1;
```

```
    }
```

```
    // Phase 2: Verify that the candidate is indeed the majority element
```

```
    count = 0;
```

```
    for (int num : nums) {
```

```
        if (num == candidate) {
```

```
            count++;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
}  
  
if (count > nums.size() / 2) {  
    return candidate;  
}  
return -1;  
}  
  
int main() {  
    vector<int> nums1 = {3, 2, 3};  
    cout << "Majority Element: " << majorityElement(nums1) << endl;  
  
    vector<int> nums2 = {2, 2, 1, 1, 1, 2, 2};  
    cout << "Majority Element: " << majorityElement(nums2) << endl;  
  
    return 0;  
}
```

OUTPUT:

```
Majority Element: 3  
Majority Element: 2  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Question 2. Single Number

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

SOLUTION:-

```
#include <vector>  
#include <iostream>  
using namespace std;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
vector<vector<int>> generate(int numRows) {
    vector<vector<int>> triangle;

    // Generate each row of Pascal's Triangle
    for (int i = 0; i < numRows; i++) {
        vector<int> row(i + 1, 1); // Initialize a row with 1s

        // Fill in the row values except the first and last element
        for (int j = 1; j < i; j++) {
            row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
        }

        triangle.push_back(row); // Add the row to the triangle
    }

    return triangle;
}

int main() {
    int numRows = 5; // Example input
    vector<vector<int>> result = generate(numRows);

    // Print Pascal's Triangle
    for (const auto& row : result) {
        for (int num : row) {
            cout << num << " ";
        }
        cout << endl;
    }
    return 0;
}
```

OUTPUT:-

```
1 2 1
1 3 3 1
1 4 6 4 1

...Program finished with exit code 0
Press ENTER to exit console.
```



Question 3 Convert Sorted Array to Binary Search Tree

Given an integer array nums where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.

SOLUTION:-

```
#include <iostream>
#include <vector>
using namespace std;
```

```
struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};
```

```
TreeNode* sortedArrayToBSTHelper(vector<int>& nums, int left, int right) {
    if (left > right) return nullptr;

    int mid = left + (right - left) / 2;
    TreeNode* root = new TreeNode(nums[mid]);
    root->left = sortedArrayToBSTHelper(nums, left, mid - 1);
    root->right = sortedArrayToBSTHelper(nums, mid + 1, right);
    return root;
}
```

```
TreeNode* sortedArrayToBST(vector<int>& nums) {
    return sortedArrayToBSTHelper(nums, 0, nums.size() - 1);
}
```

```
void preOrder(TreeNode* root) {
    if (!root) return;
    cout << root->val << " ";

    preOrder(root->left);
    preOrder(root->right);
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int main() {  
    vector<int> nums1 = {-10, -3, 0, 5, 9};  
    TreeNode* root1 = sortedArrayToBST(nums1);  
    cout << "Pre-order traversal of BST for nums1: ";  
    preOrder(root1);  
    cout << endl;  
  
    vector<int> nums2 = {1, 3};  
    TreeNode* root2 = sortedArrayToBST(nums2);  
    cout << "Pre-order traversal of BST for nums2: ";  
    preOrder(root2);  
    cout << endl;  
  
    return 0;  
}
```

OUTPUT:-

```
Pre-order traversal of BST for nums1: 0 -10 -3 5 9  
Pre-order traversal of BST for nums2: 1 3  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Q4.Merge Two Sorted Lists

You are given the heads of two sorted linked lists list1 and list2.
Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

SOLUTION:-

```
#include <iostream>  
using namespace std;  
struct ListNode {  
    int val;  
    ListNode* next;  
    ListNode(int x) : val(x), next(nullptr) {}  
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
};  
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {  
    ListNode* dummy = new ListNode(0);  
    ListNode* current = dummy;  
    while (list1 && list2) {  
        if (list1->val < list2->val) {  
            current->next = list1;  
            list1 = list1->next;  
        } else {  
            current->next = list2;  
            list2 = list2->next;  
        }  
        current = current->next;  
    }  
    if (list1) current->next = list1;  
    if (list2) current->next = list2;  
    return dummy->next;  
}  
int main() {  
    ListNode* list1 = new ListNode(1);  
    list1->next = new ListNode(2);  
    list1->next->next = new ListNode(4);  
    ListNode* list2 = new ListNode(1);  
    list2->next = new ListNode(3);  
    list2->next->next = new ListNode(4);  
    ListNode* mergedList = mergeTwoLists(list1, list2);  
    while (mergedList) {  
        cout << mergedList->val << " ";  
        mergedList = mergedList->next;  
    }  
    cout << endl;  
    return 0;  
}
```

OUTPUT:-

```
1 1 2 3 4 4  
...Program finished with exit code 0  
Press ENTER to exit console.□
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Easy

Question 1. Pascal's Triangle

Given an integer numRows, return the first numRows of Pascal's triangle.

In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:

SOLUTION:-

```
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int>> generate(int numRows) {
    vector<vector<int>> triangle(numRows);
    for (int i = 0; i < numRows; i++) {
        triangle[i].resize(i + 1, 1);
        for (int j = 1; j < i; j++)
            triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
    }
    return triangle;
}

int main() {
    int numRows = 5;
    vector<vector<int>> result = generate(numRows);
    for (auto row : result) {
        for (int num : row) cout << num << " " << endl;
    }
    return 0;
}
```

OUTPUT:-

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

...Program finished with exit code 0
Press ENTER to exit console.[]
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Question 2. Remove Element

Given an integer array `nums` sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.

Return `k`.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length
```

```
int k = removeDuplicates(nums); // Calls your implementation
```

```
assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be accepted.

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
int removeDuplicates(vector<int>& nums) {
    if (nums.empty()) return 0;
    int k = 1;
    for (int i = 1; i < nums.size(); i++) {
        if (nums[i] != nums[i - 1]) {
            nums[k] = nums[i];
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
k++;  
    }  
}  
return k;  
}  
  
int main() {  
    vector<int> nums1 = {1, 1, 2};  
    int k1 = removeDuplicates(nums1);  
    cout << k1 << endl;  
    for (int i = 0; i < k1; i++) cout << nums1[i] << " ";  
    cout << endl;  
  
    vector<int> nums2 = {0, 0, 1, 1, 1, 2, 2, 3, 3, 4};  
    int k2 = removeDuplicates(nums2);  
    cout << k2 << endl;  
    for (int i = 0; i < k2; i++) cout << nums2[i] << " ";  
    cout << endl;  
  
    return 0;  
}
```

OUTPUT:-

```
2  
1 2  
5  
0 1 2 3 4  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Question 3 Baseball Game :

You are keeping the scores for a baseball game with strange rules. At the beginning of the game, you start with an empty record.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

You are given a list of strings operations, where operations[i] is the ith operation you must apply to the record and is one of the following:

An integer x.

Record a new score of x.

'+'.

Record a new score that is the sum of the previous two scores.

'D'.

Record a new score that is the double of the previous score.

'C'.

Invalidate the previous score, removing it from the record.

Return the sum of all the scores on the record after applying all the operations.

The test cases are generated such that the answer and all intermediate calculations fit in a 32-bit integer and that all operations are valid.

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <string>
```

```
using namespace std;
```

```
int calPoints(vector<string>& ops) {  
    vector<int> record;  
    for (const string& op : ops) {  
        if (op == "C") {  
            record.pop_back();  
        } else if (op == "D") {  
            record.push_back(2 * record.back());  
        } else if (op == "+") {  
            record.push_back(record[record.size() - 1] + record[record.size() - 2]);  
        } else {  
            record.push_back(stoi(op));  
        }  
    }  
    int sum = 0;  
    for (int score : record) {  
        sum += score;  
    }  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    return sum;
}
int main() {
    vector<string> ops = {"5", "2", "C", "D", "+"};
    cout << calPoints(ops) << endl;
    return 0;
}
```

OUTPUT:-

```
30
...Program finished with exit code 0
Press ENTER to exit console.[]
```

Q5. Reverse Linked List

Given the head of a singly linked list, reverse the list, and return *the reversed list*.

SOLUTION:-

```
#include <iostream>
using namespace std;
```

```
struct ListNode {
    int val;
    ListNode* next;
    ListNode(int x) : val(x), next(nullptr) {}
};
```

```
ListNode* reverseList(ListNode* head) {
    ListNode* prev = nullptr;
    ListNode* curr = head;
    while (curr) {
        ListNode* nextNode = curr->next;
        curr->next = prev;
        prev = curr;
        curr = nextNode;
    }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
    return prev;  
}  
  
int main() {  
    ListNode* head = new ListNode(1);  
    head->next = new ListNode(2);  
    head->next->next = new ListNode(3);  
    head->next->next->next = new ListNode(4);  
  
    head = reverseList(head);  
  
    while (head) {  
        cout << head->val << " ";  
        head = head->next;  
    }  
    cout << endl;  
    return 0;  
}
```

OUTPUT:-

```
4 3 2 1  
  
...Program finished with exit code 0  
Press ENTER to exit console.█
```

Medium:

Question 1. Container With Most Water

You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `i`th line are $(i, 0)$ and $(i, \text{height}[i])$.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return the maximum amount of water a container can store.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Notice that you may not slant the container.

SOLUTION:-

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int maxArea(vector<int>& height) {
    int left = 0, right = height.size() - 1;
    int maxArea = 0;

    while (left < right) {
        int currentArea = (right - left) * min(height[left], height[right]);
        maxArea = max(maxArea, currentArea);

        if (height[left] < height[right]) {
            left++;
        } else {
            right--;
        }
    }

    return maxArea;
}

int main() {
    vector<int> height = {1, 8, 6, 2, 5, 4, 8, 3, 7};
    cout << "Maximum water container area: " << maxArea(height) << endl;
    return 0;
}
```

OUTPUT:-

```
Maximum water container area: 49

...Program finished with exit code 0
Press ENTER to exit console.
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Question 2. Valid Sudoku

Determine if a 9 x 9 Sudoku board is valid. Only the filled cells need to be validated according to the following rules:

Each row must contain the digits 1-9 without repetition.

Each column must contain the digits 1-9 without repetition.

Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.

Note:

A Sudoku board (partially filled) could be valid but is not necessarily solvable.

Only the filled cells need to be validated according to the mentioned rules.

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <unordered_set>
```

```
using namespace std;
```

```
bool isValidSudoku(vector<vector<char>>& board) {
```

```
    unordered_set<string> seen;
```

```
    for (int i = 0; i < 9; ++i) {
```

```
        for (int j = 0; j < 9; ++j) {
```

```
            char num = board[i][j];
```

```
            if (num != '.') {
```

```
                string rowKey = "row" + to_string(i) + num;
```

```
                string colKey = "col" + to_string(j) + num;
```

```
                string boxKey = "box" + to_string(i / 3) + to_string(j / 3) + num;
```

```
                if (seen.count(rowKey) || seen.count(colKey) || seen.count(boxKey)) {
```

```
                    return false;
```

```
                }
```

```
                seen.insert(rowKey);
```

```
                seen.insert(colKey);
```

```
                seen.insert(boxKey);
```

```
            }
```

```
        }
```

```
    }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

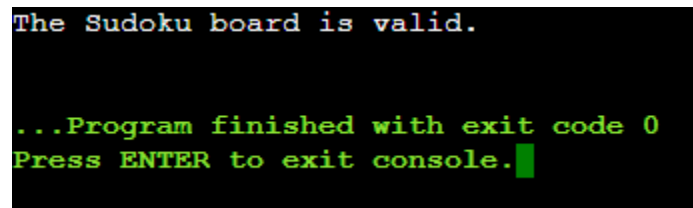
```
    return true;
}

int main() {
    vector<vector<char>>> board = {
        {'5', '3', '.', '.', '7', '.', '.', '.', '.'},
        {'6', '.', '.', '1', '9', '5', '.', '.', '.'},
        {'.', '9', '8', '.', '.', '.', '.', '6', '.'},
        {'8', '.', '.', '.', '6', '.', '.', '.', '3'},
        {'4', '.', '.', '8', '.', '3', '.', '.', '1'},
        {'7', '.', '.', '.', '2', '.', '.', '.', '6'},
        {'.', '6', '.', '.', '.', '.', '2', '8', '.'},
        {'.', '.', '.', '4', '1', '9', '.', '.', '5'},
        {'.', '.', '.', '.', '8', '.', '.', '7', '9'}
    };

    if (isValidSudoku(board)) {
        cout << "The Sudoku board is valid." << endl;
    } else {
        cout << "The Sudoku board is invalid." << endl;
    }

    return 0;
}
```

OUTPUT:-



```
The Sudoku board is valid.

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 3 : Jump Game II

You are given a 0-indexed array of integers `nums` of length `n`. You are initially positioned at `nums[0]`.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at `nums[i]`, you can jump to any `nums[i + j]` where:

$0 \leq j \leq \text{nums}[i]$ and

$i + j < n$

Return the minimum number of jumps to reach `nums[n - 1]`. The test cases are generated such that you can reach `nums[n - 1]`.

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
int jump(vector<int>& nums) {
```

```
    int n = nums.size();
```

```
    int jumps = 0, currEnd = 0, currFarthest = 0;
```

```
    for (int i = 0; i < n - 1; i++) {
```

```
        // Update the farthest reachable position from current index
```

```
        currFarthest = max(currFarthest, i + nums[i]);
```

```
        // If we've reached the end of the current jump's range
```

```
        if (i == currEnd) {
```

```
            jumps++;           // Increase the jump count
```

```
            currEnd = currFarthest; // Move to the farthest reachable point
```

```
        }
```

```
    }
```

```
    return jumps;
```

```
}
```

```
int main() {
```

```
    vector<int> nums = {2, 3, 1, 1, 4}; // Example input
```

```
    cout << "Minimum number of jumps: " << jump(nums) << endl;
```

```
    return 0;
```

```
}
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

OUTPUT:-

```
Minimum number of jumps: 2  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Q4.Populating Next Right Pointers in Each Node

You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

```
struct Node {  
    int val;  
    Node *left;  
    Node *right;  
    Node *next;  
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.

Initially, all next pointers are set to NULL.

SOLUTION:-

```
#include <iostream>  
using namespace std;
```

```
struct Node {  
    int val;  
    Node *left;  
    Node *right;  
    Node *next;  
    Node(int x) : val(x), left(NULL), right(NULL), next(NULL) {}  
};
```

```
void connect(Node* root) {  
    if (!root) return;  
  
    Node* levelStart = root;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
[
while (levelStart->left) {
    Node* current = levelStart;

    while (current) {
        current->left->next = current->right;
        if (current->next) {
            current->right->next = current->next->left;
        }
        current = current->next;
    }

    levelStart = levelStart->left;
}

int main() {
    Node* root = new Node(1);
    root->left = new Node(2);
    root->right = new Node(3);
    root->left->left = new Node(4);
    root->left->right = new Node(5);
    root->right->left = new Node(6);
    root->right->right = new Node(7);

    connect(root);
    cout << "Next of 4: " << (root->left->left->next ? root->left->left->next->val : -1) << endl;
    cout << "Next of 5: " << (root->left->right->next ? root->left->right->next->val : -1) << endl;
    cout << "Next of 6: " << (root->right->left->next ? root->right->left->next->val : -1) << endl;
    cout << "Next of 7: " << (root->right->right->next ? root->right->right->next->val : -1) << endl;
    return 0;
}
```

OUTPUT:-

```
Next of 4: 5
Next of 5: 6
Next of 6: 7
Next of 7: -1

...Program finished with exit code 0
Press ENTER to exit console.[]
```



Hard

Question 1. Maximum Number of Groups Getting Fresh Donuts

There is a donuts shop that bakes donuts in batches of `batchSize`. They have a rule where they must serve all of the donuts of a batch before serving any donuts of the next batch. You are given an integer `batchSize` and an integer array `groups`, where `groups[i]` denotes that there is a group of `groups[i]` customers that will visit the shop. Each customer will get exactly one donut.

When a group visits the shop, all customers of the group must be served before serving any of the following groups. A group will be happy if they all get fresh donuts. That is, the first customer of the group does not receive a donut that was left over from the previous group.

You can freely rearrange the ordering of the groups. Return the maximum possible number of happy groups after rearranging the groups.

SOLUTION:-

```
#include <iostream>
#include <vector>
#include <unordered_map>
using namespace std;

int maxHappyGroups(int batchSize, vector<int>& groups) {
    vector<int> count(batchSize, 0);
    for (int g : groups) {
        count[g % batchSize]++;
    }

    int result = count[0];
    int left = 0;

    for (int i = 1; i <= batchSize / 2; ++i) {
        if (i == batchSize - i) {
            result += count[i] / 2;
        } else {
            int pairs = min(count[i], count[batchSize - i]);
            result += pairs;
            count[i] -= pairs;
        }
    }

    return result + count[batchSize - 1];
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        count[batchSize - i] -= pairs;
        left += count[i];
    }
}

result += left / batchSize;
return result;
}

int main() {
    int batchSize, n;

    cout << "Enter batch size: ";
    cin >> batchSize;

    cout << "Enter the number of groups: ";
    cin >> n;

    vector<int> groups(n);
    cout << "Enter the group sizes: ";
    for (int i = 0; i < n; ++i) {
        cin >> groups[i];
    }

    int result = maxHappyGroups(batchSize, groups);
    cout << "Maximum number of happy groups: " << result << endl;

    return 0;
}
```

OUTPUT:-

```
Enter batch size: 3
Enter the number of groups: 4
Enter the group sizes: 3 3 3 6
Maximum number of happy groups: 4

...Program finished with exit code 0
Press ENTER to exit console.□
```

Question 2 Cherry Pickup II

You are given a rows x cols matrix grid representing a field of cherries where grid[i][j] represents the number of cherries that you can collect from the (i, j) cell.

You have two robots that can collect cherries for you:

Robot #1 is located at the top-left corner (0, 0), and

Robot #2 is located at the top-right corner (0, cols - 1).

Return the maximum number of cherries collection using both robots by following the rules below:

From a cell (i, j), robots can move to cell (i + 1, j - 1), (i + 1, j), or (i + 1, j + 1).

When any robot passes through a cell, It picks up all cherries, and the cell becomes an empty cell.

When both robots stay in the same cell, only one takes the cherries.

Both robots cannot move outside of the grid at any moment.

Both robots should reach the bottom row in grid.

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
int cherryPickup(vector<vector<int>>& grid) {  
    int rows = grid.size(), cols = grid[0].size();  
    vector<vector<vector<int>>> dp(rows, vector<vector<int>>(cols, vector<int>(cols, -1)));  
    dp[0][0][cols - 1] = grid[0][0] + grid[0][cols - 1];  
  
    for (int i = 1; i < rows; ++i) {  
        for (int r1 = 0; r1 < cols; ++r1) {  
            for (int r2 = 0; r2 < cols; ++r2) {  
                int maxCherries = -1;  
                for (int dr1 = -1; dr1 <= 1; ++dr1) {  
                    for (int dr2 = -1; dr2 <= 1; ++dr2) {  
                        int prevR1 = r1 + dr1, prevR2 = r2 + dr2;  
                        if (prevR1 >= 0 && prevR1 < cols && prevR2 >= 0 && prevR2 < cols) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        maxCherries = max(maxCherries, dp[i - 1][prevR1][prevR2]);

    }
}

}

if (maxCherries != -1) {
    dp[i][r1][r2] = maxCherries + grid[i][r1];
    if (r1 != r2) dp[i][r1][r2] += grid[i][r2];
}

}

}

int result = 0;
for (int r1 = 0; r1 < cols; ++r1) {
    for (int r2 = 0; r2 < cols; ++r2) {
        result = max(result, dp[rows - 1][r1][r2]);
    }
}

return result;
}

int main() {
    int rows, cols;
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> cols;
    vector<vector<int>> grid(rows, vector<int>(cols));
    cout << "Enter the grid values row by row:" << endl;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            cin >> grid[i][j];
        }
    }

    int result = cherryPickup(grid);
    cout << "Maximum number of cherries collected: " << result << endl;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    return 0;  
}
```

OUTPUT:-

```
Enter the number of rows: 3  
Enter the number of columns: 3  
Enter the grid values row by row:  
3 1 1  
1 3 5  
1 1 5  
Maximum number of cherries collected: 18  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Question 3: Maximum Number of Darts Inside of a Circular Dartboard

Alice is throwing n darts on a very large wall. You are given an array `darts` where `darts[i] = [xi, yi]` is the position of the i th dart that Alice threw on the wall.

Bob knows the positions of the n darts on the wall. He wants to place a dartboard of radius r on the wall so that the maximum number of darts that Alice throws lie on the dartboard.

Given the integer r , return the maximum number of darts that can lie on the dartboard.

SOLUTION:-

```
#include <iostream>  
#include <vector>  
#include <cmath>  
using namespace std;
```

```
double distance(double x1, double y1, double x2, double y2) {  
    return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));  
}
```

```
pair<double, double> getCircleCenter(double x1, double y1, double x2, double y2, double r) {  
    double d = distance(x1, y1, x2, y2);  
    double midX = (x1 + x2) / 2;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
double midY = (y1 + y2) / 2;
double h = sqrt(r * r - (d / 2) * (d / 2));
double dx = y2 - y1;

double dy = x1 - x2;
double len = sqrt(dx * dx + dy * dy);
dx = dx / len * h;
dy = dy / len * h;
return {midX + dx, midY + dy};
}

int maxDarts(vector<vector<int>>& darts, int r) {
    int n = darts.size();
    int result = 1;
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            double d = distance(darts[i][0], darts[i][1], darts[j][0], darts[j][1]);
            if (d > 2 * r) continue;
            pair<double, double> center = getCircleCenter(darts[i][0], darts[i][1], darts[j][0],
darts[j][1], r);
            int count = 0;
            for (int k = 0; k < n; ++k) {
                if (distance(center.first, center.second, darts[k][0], darts[k][1]) <= r) {
                    ++count;
                }
            }
            result = max(result, count);
        }
    }
    return result;
}

int main() {
    int n, r;
    cout << "Enter the number of darts: ";
    cin >> n;
    cout << "Enter the radius of the dartboard: ";
    cin >> r;
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
vector<vector<int>> darts(n, vector<int>(2));
cout << "Enter the dart positions (x y):" << endl;
for (int i = 0; i < n; ++i) {
    cin >> darts[i][0] >> darts[i][1];

}

cout << "Maximum number of darts inside the dartboard: " << maxDarts(darts, r) << endl;
return 0;
}
```

OUTPUT:-

```
Enter the number of darts: 5
Enter the radius of the dartboard: 2
Enter the dart positions (x y):
1 1
2 2
3 3
4 4
5 5
Maximum number of darts inside the dartboard: 3

...Program finished with exit code 0
Press ENTER to exit console.
```

Very Hard

Question 1. Find Minimum Time to Finish All Jobs

You are given an integer array jobs, where jobs[i] is the amount of time it takes to complete the ith job. There are k workers that you can assign jobs to. Each job should be assigned to exactly one worker. The working time of a worker is the sum of the time it takes to complete all jobs assigned to them. Your goal is to devise an optimal assignment such that the maximum working time of any worker is minimized.

Return the minimum possible maximum working time of any assignment.

SOLUTION:-



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
using namespace std;

bool canAssignJobs(const vector<int>& jobs, int k, int maxTime) {
    int workers = 1, currentTime = 0;
    for (int job : jobs) {
        if (currentTime + job > maxTime) {
            workers++;
            currentTime = job;
            if (workers > k) return false;
        } else {
            currentTime += job;
        }
    }
    return true;
}

int minimumTimeToFinishJobs(vector<int>& jobs, int k) {
    int left = *max_element(jobs.begin(), jobs.end());
    int right = accumulate(jobs.begin(), jobs.end(), 0);
    int result = right;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (canAssignJobs(jobs, k, mid)) {
            result = mid;
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }

    return result;
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int main() {  
    int n, k;  
  
    cout << "Enter the number of jobs: ";  
    cin >> n;  
  
    vector<int> jobs(n);  
  
    cout << "Enter the job times: ";  
    for (int i = 0; i < n; ++i) {  
        cin >> jobs[i];  
    }  
  
    cout << "Enter the number of workers: ";  
    cin >> k;  
  
    int result = minimumTimeToFinishJobs(jobs, k);  
  
    cout << "Minimum maximum working time: " << result << endl;  
  
    return 0;  
}
```

OUTPUT:-

```
Enter the number of jobs: 4  
Enter the job times: 6 2 5 8  
Enter the number of workers: 3  
Minimum maximum working time: 8  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Question 2. Minimum Number of People to Teach

On a social network consisting of m users and some friendships between users, two users can communicate with each other if they know a common language.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

You are given an integer n , an array `languages`, and an array `friendships` where:

There are n languages numbered 1 through n ,

`languages[i]` is the set of languages the i th user knows, and

`friendships[i] = [ui, vi]` denotes a friendship between the users ui and vi .

You can choose one language and teach it to some users so that all friends can communicate with each other. Return the minimum number of users you need to teach.

Note that friendships are not transitive, meaning if x is a friend of y and y is a friend of z , this doesn't guarantee that x is a friend of z .

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <unordered_set>
```

```
#include <unordered_map>
```

```
using namespace std;
```

```
int minimumNumberOfPeopleToTeach(int n, vector<vector<int>>& languages,  
vector<vector<int>>& friendships) {
```

```
    unordered_set<int> noCommonLanguage;
```

```
    for (const auto& friendship : friendships) {
```

```
        int u = friendship[0] - 1;
```

```
        int v = friendship[1] - 1;
```

```
        unordered_set<int> setU(languages[u].begin(), languages[u].end());
```

```
        bool common = false;
```

```
        for (int lang : languages[v]) {
```

```
            if (setU.count(lang)) {
```

```
                common = true;
```

```
                break;
```

```
            }
```

```
        }
```

```
        if (!common) {
```

```
            noCommonLanguage.insert(u);
```

```
            noCommonLanguage.insert(v);
```

```
        }
```

```
    }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
unordered_map<int, int> langCount;
for (int user : noCommonLanguage) {
    for (int lang : languages[user]) {
        langCount[lang]++;
    }
}

int maxCommonLang = 0;
for (const auto& [lang, count] : langCount) {
    maxCommonLang = max(maxCommonLang, count);
}

return noCommonLanguage.size() - maxCommonLang;
}

int main() {
    int n, m, f;
    cout << "Enter the number of languages: ";
    cin >> n;
    cout << "Enter the number of users: ";
    cin >> m;

    vector<vector<int>> languages(m);
    cout << "Enter the languages each user knows (end each list with -1):" << endl;
    for (int i = 0; i < m; ++i) {
        cout << "User " << i + 1 << ": ";
        int lang;
        while (cin >> lang && lang != -1) {
            languages[i].push_back(lang);
        }
    }

    cout << "Enter the number of friendships: ";
    cin >> f;
    vector<vector<int>> friendships(f, vector<int>(2));
    cout << "Enter the friendships (two space-separated user indices per line):" << endl;
    for (int i = 0; i < f; ++i) {
        cin >> friendships[i][0] >> friendships[i][1];
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}

int result = minimumNumberOfPeopleToTeach(n, languages, friendships);
cout << "Minimum number of users to teach: " << result << endl;

return 0;
}
```

OUTPUT:-

```
Enter the number of users: 3
Enter the languages each user knows (end each list with -1):
User 1: 1 2 -1
User 2: 3 -1
User 3: 2 3 -1
Enter the number of friendships: 3
Enter the friendships (two space-separated user indices per line):
1 2
1 3
2 3
Minimum number of users to teach: 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 3 Count Ways to Make Array With Product

You are given a 2D integer array, queries. For each queries[i], where queries[i] = [ni, ki], find the number of different ways you can place positive integers into an array of size ni such that the product of the integers is ki. As the number of ways may be too large, the answer to the ith query is the number of ways modulo $10^9 + 7$.

Return an integer array answer where answer.length == queries.length, and answer[i] is the answer to the ith query.

Example 1:

Input: queries = [[2,6],[5,1],[73,660]]

Output: [4,1,50734910]



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Explanation: Each query is independent.

[2,6]: There are 4 ways to fill an array of size 2 that multiply to 6: [1,6], [2,3], [3,2], [6,1].

[5,1]: There is 1 way to fill an array of size 5 that multiply to 1: [1,1,1,1,1].

[73,660]: There are 1050734917 ways to fill an array of size 73 that multiply to 660.

1050734917 modulo $10^9 + 7 = 50734910$.

Example 2:

Input: queries = [[1,1],[2,2],[3,3],[4,4],[5,5]]

Output: [1,2,3,10,5]

Constraints:

$1 \leq \text{queries.length} \leq 104$

$1 \leq n_i, k_i \leq 104$

Q4 Maximum Twin Sum of a Linked List

In a linked list of size n , where n is **even**, the i^{th} node (**0-indexed**) of the linked list is known as the **twin** of the $(n-1-i)^{\text{th}}$ node, if $0 \leq i \leq (n/2) - 1$.

- For example, if $n = 4$, then node 0 is the twin of node 3, and node 1 is the twin of node 2. These are the only nodes with twins for $n = 4$.

The **twin sum** is defined as the sum of a node and its twin.

Given the head of a linked list with even length, return *the maximum twin sum of the linked list*.

SOLUTION:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
struct ListNode {
```

```
    int val;
```

```
    ListNode* next;
```

```
    ListNode(int x) : val(x), next(nullptr) {}
```

```
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
ListNode* reverseList(ListNode* head) {
    ListNode* prev = nullptr;
    ListNode* curr = head;
    while (curr) {
        ListNode* nextNode = curr->next;
        curr->next = prev;
        prev = curr;
        curr = nextNode;
    }
    return prev;
}

int pairSum(ListNode* head) {
    vector<int> vals;
    ListNode* curr = head;
    while (curr) {
        vals.push_back(curr->val);
        curr = curr->next;
    }

    int n = vals.size();
    int maxSum = 0;
    for (int i = 0; i < n / 2; ++i) {
        maxSum = max(maxSum, vals[i] + vals[n - 1 - i]);
    }
    return maxSum;
}

int main() {
    ListNode* head = new ListNode(5);
    head->next = new ListNode(4);
    head->next->next = new ListNode(2);
    head->next->next->next = new ListNode(1);

    cout << pairSum(head) << endl;
    return 0;
}
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

OUTPUT:-

```
6
...Program finished with exit code 0
Press ENTER to exit console.
```

Q5. Insert Greatest Common Divisors in Linked List

Given the head of a linked list head, in which each node contains an integer value.

Between every pair of adjacent nodes, insert a new node with a value equal to the **greatest common divisor** of them.

Return *the linked list after insertion*.

The **greatest common divisor** of two numbers is the largest positive integer that evenly divides both numbers.

SOLUTION:-

```
#include <iostream>
using namespace std;
```

```
struct ListNode {
    int val;
    ListNode* next;
    ListNode(int x) : val(x), next(nullptr) {}
};
```

```
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

```
ListNode* insertGCD(ListNode* head) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
[
    ListNode* curr = head;
    while (curr != nullptr && curr->next != nullptr) {
        int gcdValue = gcd(curr->val, curr->next->val);
        ListNode* newNode = new ListNode(gcdValue);
        newNode->next = curr->next;
        curr->next = newNode;
        curr = newNode->next;
    }
    return head;
}

void printList(ListNode* head) {
    while (head != nullptr) {
        cout << head->val << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    int n;
    cout << "Enter the number of nodes: ";
    cin >> n;

    int val;
    cout << "Enter the values for the nodes: ";
    cin >> val;
    ListNode* head = new ListNode(val);
    ListNode* curr = head;

    for (int i = 1; i < n; ++i) {
        cin >> val;
        curr->next = new ListNode(val);
        curr = curr->next;
    }

    head = insertGCD(head);
    printList(head);
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

[

```
    return 0;  
}
```

OUTPUT:-

```
Enter the number of nodes: 4  
Enter the values for the nodes: 6 12 15 30  
6 6 12 3 15 15 30  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```