



## DAY – 2 Array & Linked List

Name: Daksh

UID: 22BCS15372

Section: IOT-620(B)

### 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.

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

```
int findMajorityElement(int arr[],
int n) {    int count = 0, candidate =
-1;    for (int i = 0; i < n; i++) {
if (count == 0) {        candidate
= arr[i];
count = 1;
    }
else if (arr[i] ==
candidate) {
count++;        } else {
count--;
    }
}
```

```
    count = 0;    for (int i = 0; i < n;
i++) {        if (arr[i] == candidate)
{            count++;
        }
    }
```

```
    }

    if (count > n / 2) {
        return candidate;    }
    else {
        return -1; // No majority element found
    }
}

int main() {
    int nums1[] = {3, 2, 3};    int n1 = sizeof(nums1) /
sizeof(nums1[0]);    int majorityElement1
= findMajorityElement(nums1, n1);    cout << "Example 1 Output: "
<< majorityElement1 << endl;

    int nums2[] = {2, 2, 1, 1, 1, 2, 2};    int n2 = sizeof(nums2) /
sizeof(nums2[0]);    int majorityElement2 =
findMajorityElement(nums2, n2);    cout << "Example 2 Output: "
<< majorityElement2 << endl;

    return 0;
}
```

```
Example 1 Output: 3
Example 2 Output: 2
```

## 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.

```
#include <iostream>
```

```
using namespace std;

int singleNumber(int nums[], int n)
{   int result = 0;   for (int i = 0; i
<
n; i++) {
    result ^= nums[i];
}
return result;
}

int main() {
    int nums1[] = {2, 2, 1};   int n1 = sizeof(nums1) /
sizeof(nums1[0]);   int single1 = singleNumber(nums1, n1);
cout << "Example 1 Output: " << single1 << endl;

    int nums2[] = {4, 1, 2, 1, 2};   int n2 = sizeof(nums2) /
sizeof(nums2[0]);   int single2 = singleNumber(nums2, n2);
cout << "Example 2 Output: " << single2 << endl;

    int nums3[] = {1};   int n3 = sizeof(nums3) /
sizeof(nums3[0]);   int single3 = singleNumber(nums3, n3);
cout << "Example 3 Output: " << single3 << endl;

    return 0;
}
```

```
Example 1 Output: 1
Example 2 Output: 4
Example 3 Output: 1
```



## Easy

### Question 1. Pascal's Triangle

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

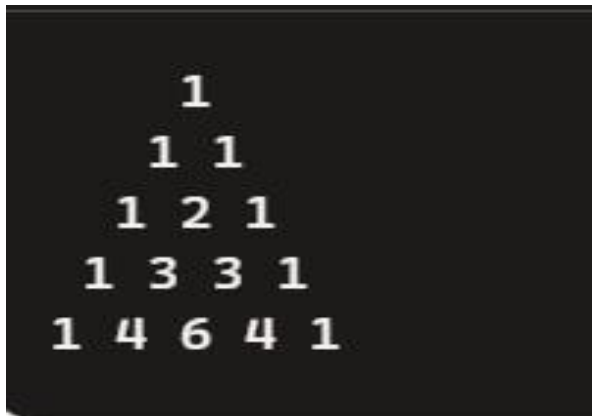
In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:

#### Code:

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

void printPascal(int n) {
    for (int line = 0; line < n; line++) {
        for (int space = 0; space < n - line - 1; space++) {
            cout << " ";
        }
        int value = 1;
        for (int i = 0; i <= line; i++)
        {
            cout << value << "
";
            value = value * (line - i) / (i + 1);
        }
        cout << endl;
    }
}

int main() {
    int
    numRows;
    cout << "Enter the number of rows:
";
    cin >> numRows;
    printPascal(numRows);
    return 0;
}
```



## Question 2. Remove Element

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

int removeDuplicates(int nums[],
int n) {    if (n == 0) return 0;
int k = 1;    for (int i = 1; i < n;
i++) {
if (nums[i] != nums[i - 1]) {
nums[k] =
nums[i];
k++;    }
}    return k;
}

int main() {    int nums1[] = {1, 1, 2};
int n1 = sizeof(nums1) /
sizeof(nums1[0]);    int k1 =
removeDuplicates(nums1, n1);
    cout << "Example 1 Output: " << k1 << ", nums = [";    for
(int i = 0; i < k1; i++) {        cout << nums1[i];        if (i < k1
- 1) cout
<< ", ";
    }
    cout << ", _]" << endl;

    int nums2[] = {0, 0, 1, 1, 1, 2, 2, 3, 3, 4};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int n2 = sizeof(nums2) / sizeof(nums2[0]);
int k2 = removeDuplicates(nums2, n2);
cout << "Example 2 Output: " << k2 << ", nums = ["; for
(int
i = 0; i < k2; i++) {      cout << nums2[i];
    if (i < k2 - 1) cout << ", ";
}
cout << ", _]" << endl;

return 0;
}
```

```
Example 1 Output: 2, nums = [1, 2, _]
Example 2 Output: 5, nums = [0, 1, 2, 3, 4, _]

=== Code Execution Successful ===
```

## Q3 .Remove Linked List Elements

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

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

ListNode* removeElements(ListNode* head, int val) {
    while (head != NULL && head->val == val) {
        ListNode* temp =
        head; head = head->next;
        delete temp;
    }

    ListNode* current = head;
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        while (current != NULL && current->next !=
NULL) {            if (current->next->val == val) {
ListNode* temp = current->next;            current-
>next = current->next->next;
                delete temp;
        } else {
                current = current->next;
        }
    }

    return head;
}

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

int main() {
    ListNode* head = new ListNode(1);    head-
>next = new ListNode(2);    head->next->next =
new ListNode(6);    head->next->next->next = new
ListNode(3);    head->next->next->next->next =
new ListNode(4);    head->next->next->next-
>next->next = new ListNode(5);
    head->next->next->next->next->next->next = new
ListNode(6);

    cout << "Original List: ";
    printList(head);    int val = 6;
    head = removeElements(head,
val);

    cout << "List after removing " << val << ": ";
    printList(head);

    return 0;
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
Original List: 1 2 6 3 4 5 6  
List after removing 6: 1 2 3 4 5
```

## Medium

### Question 1. Container With Most Water

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

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

    while (left < right) {
int width = right - left;
        int current_area = min(height[left], height[right]) *
width;        max_area = max(max_area, current_area);

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

    return max_area;
}

int main() {
    vector<int> height1 = {1, 8, 6, 2, 5, 4, 8, 3, 7};    cout << "Example 1
Output: " << maxArea(height1) << endl;

    vector<int> height2 = {1, 1};    cout << "Example 2
Output: " << maxArea(height2) << endl;
```



```
    return 0;  
}
```

```
Output  
Example 1 Output: 49  
Example 2 Output: 1  
  
=== Code Execution Successful ===
```

## Question 2. Valid Sudoku

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

```
bool isValidSudoku(vector<vector<char>>& board) {    vector<unordered_set<char>> rows(9),  
cols(9), boxes(9);
```

```
    for (int i = 0; i < 9;  
i++) {        for (int j = 0;  
j < 9; j++) {            char  
num = board[i][j];  
if (num == '.') continue;
```

```
        int boxIndex = (i / 3) * 3 + j / 3;  
        if (rows[i].count(num) || cols[j].count(num) || boxes[boxIndex].count(num)) {  
return false;  
        }
```

```
        rows[i].insert(num);  
cols[j].insert(num);  
boxes[boxIndex].insert(num);
```

```

    }
}

return true;
}

int main() {
vector<vector<char>>> board1
= {
    {'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'}
};
cout << "Example 1 Output: " << (isValidSudoku(board1) ? "true" : "false") << endl;

return 0;

```

Output

Example 1 Output: true  
  
=== Code Execution Successful ===

## Question 3 : Jump Game II

```

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

int jump(vector<int>& nums) {

```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int n =  
nums.size();  
if (n == 1) return  
0;  
  
    int jumps = 0, currentEnd = 0, farthest = 0;  
    for (int i = 0; i < n - 1; i++) {  
        farthest = max(farthest, i + nums[i]);  
        if (i == currentEnd) {  
            jumps++;  
            currentEnd = farthest;  
        }  
    }  
  
    return jumps;  
}  
  
int main() {  
    vector<int> nums1 = {2, 3, 1, 1, 4};    cout <<  
"Example 1 Output: " << jump(nums1) << endl;  
  
    vector<int> nums2 = {2, 3, 0, 1, 4};    cout <<  
"Example 2 Output: " << jump(nums2) << endl;  
  
    return 0;  
}
```

## Output

```
Example 1 Output: 2  
Example 2 Output: 2
```

```
=== Code Execution Successful ===|
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 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.

#### Code:

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

int maxHappyGroups(int batchSize, vector<int>& groups) {
    unordered_map<int, int> remainderCount;
    for (int group : groups) {
        remainderCount[group % batchSize]++;
    }

    int happyGroups = remainderCount[0];
    for (int i = 1; i <= batchSize / 2; i++) {
        if (i == batchSize - i) {
            happyGroups += remainderCount[i] / 2;
        } else {
            happyGroups += min(remainderCount[i], remainderCount[batchSize - i]);
        }
    }

    return happyGroups;
}
```

```
int main() {  
    int batchSize = 3;  
  
    vector<int> groups = {1, 2, 3, 4, 5, 6};    cout << "Maximum number of happy groups: "  
    << maxHappyGroups(batchSize, groups) << endl;  
  
    return 0;  
}
```

Maximum number of happy groups: 3

## Ques 2 :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  $(n-1-i)$ 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*.

## Code :

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



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int pairSum(ListNode* head) {
    vector<int> values;    ListNode* current
    = head;    while (current != NULL) {
        values.push_back(current->val);
        current = current->next;
    }

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

    return maxSum;
}

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

int main() {
    ListNode* head1 = new ListNode(5);    head1->next = new
    ListNode(4);    head1->next->next = new ListNode(2);    head1-
    >next->next->next = new ListNode(1);    cout <<
    "Example 1 Output: " << pairSum(head1) << endl;

    ListNode* head2 = new ListNode(4);    head2->next = new
    ListNode(2);    head2->next->next = new ListNode(2);    head2-
    >next->next->next = new ListNode(3);
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    cout << "Example 2 Output: " << pairSum(head2) << endl;  
  
    return 0;  
}
```

## Output

```
Example 1 Output: 6  
Example 2 Output: 7
```

```
=== Code Execution Successful ===|
```