



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

DAY 2

Student Name: Suman

UID: 22BCS15488

Branch: BE-CSE

Section/Group: 620 - B

Date of Performance: 20/12/24

Problem 1

1. Aim: Majority Elements
2. Problem Statement: 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.

3. Code:

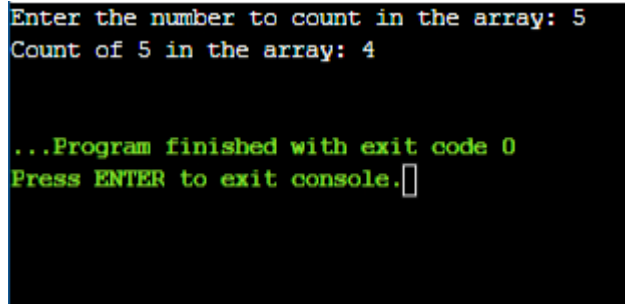
```
#include <iostream>
using namespace std;

int main() {
    int arr[11] = {1, 2, 3, 4, 5,
6, 3, 4, 5, 5, 5};
    int value, count = 0;
    cout << "Enter the
number to count in the
array: ";
    cin >> value;

    for (int i = 0; i < 11;
i++) {
        if (arr[i] == value) {
            count++;
        }
    }
}
```

```
    }  
    cout << "Count of " <<  
value << " in the array: " <<  
count << endl;  
    return 0;  
}
```

4. Output:



```
Enter the number to count in the array: 5  
Count of 5 in the array: 4  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem 2

1. Aim: Pascal's Triangle

2. Problem Statement: 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.

3. Code:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    int rows;
```

```
    cout << "Enter the number of rows for Pascal's Triangle: ";
```

```
    cin >> rows;
```

```
for (int i = 0; i < rows; i++) {  
    int value = 1;  
    for(int j=0;j<rows-i-1;j++){  
        cout<<" ";  
    }  
    for (int j = 0; j <= i; j++) {  
        cout << value << " ";  
        value = value * (i - j) / (j + 1);  
    }  
    cout << endl;  
}  
return 0;  
}
```

4. Output:

```
Enter the number of rows for Pascal's Triangle: 6  
    1  
  1 1  
 1 2 1  
1 3 3 1  
1 4 6 4 1  
1 5 10 10 5 1  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem 3

1. Aim: Single Number



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

2. Problem Statement: 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.

3. Code:

```
#include <iostream>

#include <vector>

using namespace std;

int singleNumber(vector<int>& nums) {
    int result = 0;
    for (int num : nums) {
        result ^= num;
    }
    return result;
}

int main() {
    int n;
    cout << "Enter the number of elements in the array: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
  
int single = singleNumber(nums);  
  
cout << "The single number is: " << single << endl;  
  
return 0;  
  
}
```

4. Output:

```
The single number is: 1
```

Problem 4

1. Aim: Merge Two Sorted Lists
2. Problem Statement: 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

3. Code:

```
#include<iostream>  
  
using namespace std;  
  
struct ListNode {  
  
    int val;  
  
    ListNode *next;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

};

ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {

    ListNode dummy(0);

    ListNode* tail = &dummy;

    while (list1 != nullptr && list2 != nullptr)
    {
        if (list1->val < list2->val) {
            tail->next = list1;
            list1 = list1->next;
        } else
        {
            tail->next = list2;
            list2 = list2->next;
        }
        tail = tail->next;
    }

    if (list1 != nullptr) {
        tail->next = list1;
    } else
    {
        tail->next = list2;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }  
    return dummy.next;  
}  
void printList(ListNode* head) {  
    while (head != nullptr) {  
        cout << head->val << " ";  
        head = head->next;  
    }    cout << endl;  
}  
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);  
    printList(mergedList);  
    return 0;  
}
```

4. Output:

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

Problem 5

1. Aim: Remove Linked List Elements
2. Problem Statement: Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return the new head.

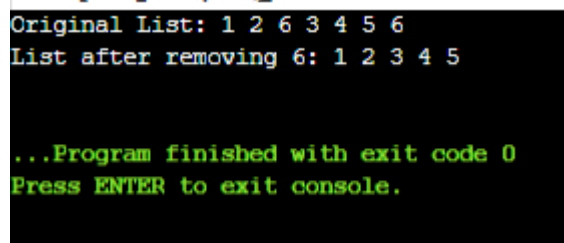
3. Code:

```
#include <iostream>  
using namespace std;  
struct ListNode {  
    int val;  
    ListNode* next;  
    ListNode(int value) : val(value), next(nullptr) { }  
};  
ListNode* removeElements(ListNode* head, int val) {  
    ListNode* dummy = new ListNode(0);  
    dummy->next = head;  
    ListNode* current = dummy;  
    while (current->next != nullptr) {  
        if (current->next->val == val) {  
            ListNode* nodeToDelete = current->next;  
            current->next = current->next->next;  
            delete nodeToDelete;  
        } else {  
            current = current->next; // Move to the next node  
        }  
    }  
    return dummy->next;  
}
```



```
    }  
}  
  
ListNode* newHead = dummy->next;  
delete dummy;  
return newHead;  
}  
void printList(ListNode* head) {  
    while (head != nullptr) {  
        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;  
}
```

4. Output:



```
Original List: 1 2 6 3 4 5 6  
List after removing 6: 1 2 3 4 5  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```



Problem 6

1. Aim: Reverse Linked List
2. Problem Statement: Given the head of a singly linked list, reverse the list, and return the reversed list.

3. Code:

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

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

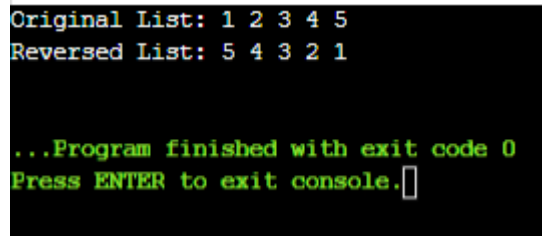
ListNode* reverseList(ListNode* head) {
    ListNode* prev = nullptr;
    ListNode* curr = head;

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

void printList(ListNode* head) {
    while (head != nullptr) {
        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(3);  
    head->next->next->next = new ListNode(4);  
    head->next->next->next->next = new ListNode(5);  
  
    cout << "Original List: ";  
    printList(head);  
    head = reverseList(head);  
    cout << "Reversed List: ";  
    printList(head);  
  
    return 0;  
}
```

4. Output:



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

Problem 7

1. Aim: Reverse a Number
2. Problem Statement: Reverse the digits of a given number n.
For example, if the input number is 12345, the output should be 54321. The task involves using loops and modulus operators to extract the digits and construct the reversed number

3. Task: Given an integer n, print the number with its digits in reverse order.

4. Code:

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

int main() {    int num, rev_num = 0;
cout << "Enter a number: ";    cin >> num;
while (num != 0) {        rev_num =
rev_num * 10 + num % 10;        num /=
10;
    }
    cout << "Reversed Number: " << rev_num << endl;
return 0;
}
```

5. Output:

```
Original list: 1 2 3 4 5
Reversed list: 5 4 3 2 1
```

Problem 8

1. Aim: Container With Most Water

2. Problem Statement: You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, 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

3. Code:

```
#include <iostream>
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

int maxArea(vector<int>& height) {
    int left = 0, right = height.size() - 1;
    int maxWater = 0;
    while (left < right) {
        int currentWater = (right - left) * min(height[left], height[right]);
        maxWater = max(maxWater, currentWater);
        if (height[left] < height[right]) {
            left++;
        } else {
            right--;
        }
    }
    return maxWater;
}

int main() {
    int n;
    cout << "Enter the number of elements in the height array: ";
    cin >> n;
    vector<int> height(n);
    cout << "Enter the heights of the lines: ";
    for (int i = 0; i < n; i++) {
        cin >> height[i];
    }
    int result = maxArea(height);
    cout << "The maximum amount of water the container can store is: "
    << result << endl;
    return 0;
}
```

4. Output:

```
Enter the number of elements in the height array: 4
Enter the heights of the lines: 3
6
7
8
The maximum amount of water the container can store is: 12

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

Problem 9

1. Aim: Jump Game II.
2. Problem Statement: You are given a 0-indexed array of integers `nums` of length `n`. You are initially positioned at `nums[0]`. 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]`.

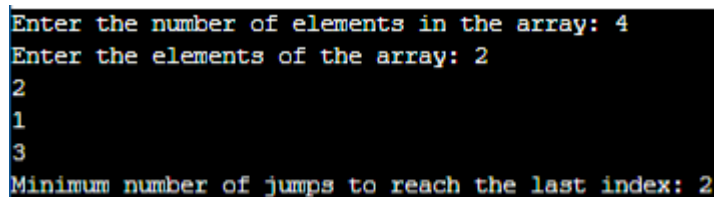
3. Code:

```
#include <iostream>
#include <vector>
using namespace std;
int minJumps(vector<int>& nums) {
    int n = nums.size();
    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;
        if (currentEnd >= n - 1) break;
    }
}
return jumps;
}
int main() {
    int n;
    cout << "Enter the number of elements in the array: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
    }
    int result = minJumps(nums);
    cout << "Minimum number of jumps to reach the last index: " << result <<
endl;

    return 0;
}
```

4. Output:



```
Enter the number of elements in the array: 4
Enter the elements of the array: 2
2
1
3
Minimum number of jumps to reach the last index: 2
```

Problem 10

1. Aim: Linked List Cycle.
2. Problem Statement: 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. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

3. Code:

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

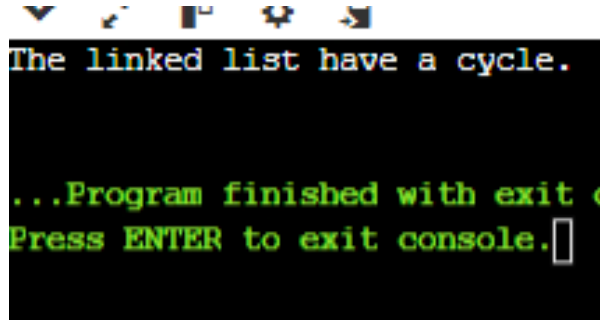
bool hasCycle(ListNode *head)
{ if (head == nullptr) return false;
  ListNode *slow = head;
  ListNode *fast = head;
  while (fast != nullptr && fast->next != nullptr)
  { slow = slow->next;
    fast = fast->next->next;
    if (slow == fast) { return true;
                      }
  }
  return false;
}

int main() {
  ListNode *head = new ListNode(3);
  ListNode *second = new ListNode(2);
  ListNode *third = new ListNode(0);
  ListNode *fourth = new ListNode(-4);
  head->next = second; second->next = third;
```



```
third->next = fourth;  
fourth->next = second;  
if (hasCycle(head)) {  
    std::cout << "The linked list has a cycle." << std::endl;  
} else {  
    std::cout << "The linked list does not have a cycle." << std::endl;  
}  
return 0;  
}
```

4. Output:



```
The linked list have a cycle.  
  
...Program finished with exit c  
Press ENTER to exit console.
```