Write a program for finding square root of a number

Input : 4
Output : 2

Input : 64
Output : 8

```
while (low <= high) {
    int mid = low + (high - low) / 2;
    int square = mid * mid;

    if (square == x) {
        return mid; // Return mid if square is equal to x
    } else if (square < x) {
        result = mid; // Update result and search in the right half
        low = mid + 1;
    } else {
        high = mid - 1; // Search in the left half
    }
}

return result;
```

For More Understanding :
BS-10. Finding Sqrt of a number using Binary Search

Write A program for finding occurrence of each character in input String

Input : “code bashers”
Output : c=1,o=1,d=1,e=2,b=1,a=1,s=2,h=1,r=1

```
void countCharacters(const string& str) {
    unordered_map<char, int> charCount; // Hashmap to store character counts

    // Iterate through the string and update character counts
    for (char c : str) {
        // If character already exists in the hashmap, increment its count
        if (charCount.find(c) != charCount.end()) {
            charCount[c]++;
        } else {
            // If character doesn't exist, initialize its count to 1
            charCount[c] = 1;
        }
    }

    // Print the character counts
    for (const auto& pair : charCount) {
        cout << "'" << pair.first << "' occurs " << pair.second << " times" << endl;
    }
}
```

For More Understanding:
Frequently Asked Java Program 26: How To Count Occurrences of a …

Sort The array Containing 0s , 1s and 2s

Input : [1 0 2 2 0 1 0]
Output : 0 0 0 1 1 2 2

```
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]);
            low++;
            mid++;
        } else if (nums[mid] == 1) {
            mid++;
        } else {
            swap(nums[mid], nums[high]);
            high--;
        }
    }
}
```

For more understanding :
Sort an array of 0's 1's & 2's | Intuition of Algo🔥 | C++ Java …

Find Maximum Sum Sub Array (kadanes Algorithm)

Input : [1 -2 3 4]
Output : 7 (3+4)

```
int maxSubArray(vector<int>& nums) {
    int max_sum = nums[0];
    int current_sum = nums[0];

    for (int i = 1; i < nums.size(); ++i) {
        current_sum = max(nums[i], current_sum + nums[i]);
        max_sum = max(max_sum, current_sum);
    }

    return max_sum;
}
```

For More Understanding :
Kadane's Algo in 16 minutes || Algorithms for Placements

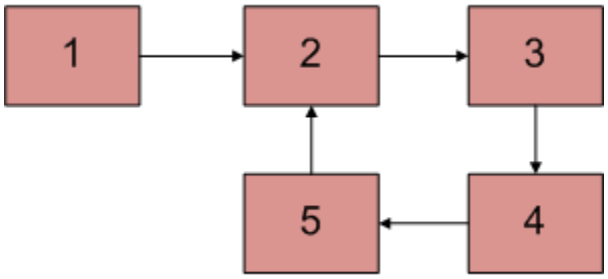
Find the element that appears once in the array and rest all appears twice

Input : [1 1 2 3 3 4 4]
Output : 2

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

For More Understanding :
Solving Arrays Questions | Find the element that appears on…

Find Loop In the Linked List




```
bool hasCycle(ListNode *head) {
    if (!head || !head->next) {
        return false; // No cycle if list has 0 or 1 node
    }

    ListNode *slow = head;
    ListNode *fast = head->next;

    while (slow != fast) {
        if (!fast || !fast->next) {
            return false; // No cycle if fast pointer reaches end of list
        }
        slow = slow->next;
        fast = fast->next->next;
    }

    return true; // Cycle detected if slow and fast pointers meet
}
```

For more Understanding :

Linked list cycle Approach | Leetcode 141 | easily explained …

Find Middle Of Linked List

Input : 1 -> 2 -> 3 -> 4 -> 5
Output : 3

```
ListNode* findMiddle(ListNode* head) {
    if (!head) {
        return nullptr; // Return nullptr if list is empty
    }

    ListNode *slow = head;
    ListNode *fast = head;

    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
    }

    return slow; // Return the middle node
}
```

For More Understanding :

Middle of linked list leetcode C++ | Leetcode 876 |Tortoise a…

Reverse Linked List

Input : 1 -> 2 -> 3 -> 4 -> 5
Output : 5 -> 4 -> 3 -> 2 -> 1

```
ListNode* reverseList(ListNode* head) {
    ListNode *prev = nullptr;
    ListNode *current = head;

    while (current) {
        ListNode *nextNode = current->next;
        current->next = prev;
        prev = current;
        current = nextNode;
    }

    return prev;
}
```

For better Understanding :

Reverse Linked List Leetcode | Leetcode 206 | Iterative Appr…

Find next greater element in Array

Input : 1 2 3 4 5
Output : 2 3 4 5 -1

```
vector<int> nextGreaterElement(vector<int>& nums) {
    int n = nums.size();
    vector<int> result(n, -1);
    stack<int> st;

    for (int i = n - 1; i >= 0; i--) {
        while (!st.empty() && st.top() <= nums[i]) {
            st.pop();
        }
        if (!st.empty()) {
            result[i] = st.top();
        }
        st.push(nums[i]);
    }

    return result;
}
```

For better Understanding :

Next Greater Element | Two Variants | Leetcode

Reverse words In string

Input : code bashers
Output : bashers code

```
void reverseWords(string& s) {
    // Step 1: Reverse the entire string
    reverse(s.begin(), s.end());

    // Step 2: Reverse each word in the reversed string
    int start = 0, end = 0;
    while (end < s.length()) {
        while (end < s.length() && s[end] != ' ') {
            end++;
        }
        reverse(s.begin() + start, s.begin() + end);
        start = end + 1;
        end = start;
    }
}
```

For better understanding :
https://www.geeksforgeeks.org/reverse-words-in-a-given-string/

Implement Stack using Array / Queue Using Array

3.2 Implementation of Stack using Array | Data Structure an…

4.2 Implementation of Queue using Arrays | Data Structures …

Searching Algorithms

Linear Search :

Linear Search Algorithm in Hindi | Linear search in C++ | Dat…

Binary Search :

Binary Search Algorithm Explained | Leetcode 704 | Leetcod…

Sorting Algorithms :

Bubble sort , Selection Sort , Insertion Sort , Quick Sort and Merge Sort

Link For Playlist :
https://www.youtube.com/playlist?list=PLuZ_bd9XlByzTIP5j1aWXo7smCIxvzd2D

Code to swap two variable without using the third variable.

```
# Initial values
a = 5
b = 10

# Swapping using arithmetic operations
a = a + b
b = a - b
a = a - b

print("After swapping: a =", a, "b =", b)
```

To check whether a number is prime or not ?

```
bool isPrime(int n)
{
    // Corner case
    if (n <= 1)
        return false;

    // Check from 2 to square root of n
    for (int i = 2; i <= sqrt(n); i++)
        if (n % i == 0)
            return false;

    return true;
}
```

Code To print pyramid ?

```
*

**

***

****

*****
```

```
void printRightAngleTriangle(int height) {
    for (int i = 1; i <= height; ++i) {
        // Print stars for the current row
        for (int j = 1; j <= i; ++j) {
            cout << "*";
        }
        // Move to the next line
        cout << endl;
    }
}
```

Check if string is palindrome or not

```
bool isPalindrome(const std::string& str) {
    int left = 0;
    int right = str.length() - 1;

    while (left < right) {
        if (str[left] != str[right]) {
            return false;
        }
        left++;
        right--;
    }
    return true;
}
```

Remove duplicate from array

```
void removeDuplicates(int arr[], int &n) {
    if (n == 0 || n == 1) {
        return;
    }

    std::sort(arr, arr + n);

    int index = 1;

    for (int i = 1; i < n; ++i) {
        if (arr[i] != arr[i - 1]) {
            arr[index++] = arr[i];
        }
    }
    n = index;
}
```

Reverse of a number program

```
int reverseNumber(int num) {
    int reversed = 0;
    while (num != 0) {
        int digit = num % 10;
        reversed = reversed * 10 + digit;
        num /= 10;
    }
    return reversed;
}
```

Multiply number with 2 without using * operator

```
int multiplyByTwo(int num) {
    return num << 1;
}
```

Find the 2nd largest element in an array

```
int secondLargest(int arr[], int size) {
    // Initialize variables to store the largest and second largest elements
    int largest = INT_MIN;
    int secondLargest = INT_MIN;

    // Traverse the array to find the largest and second largest elements
    for (int i = 0; i < size; ++i) {
        if (arr[i] > largest) {
            secondLargest = largest; // Update second largest to previous largest
            largest = arr[i]; // Update largest to current element
        } else if (arr[i] > secondLargest && arr[i] != largest) {
            secondLargest = arr[i]; // Update second largest
        }
    }

    return secondLargest;
}
```