Top 100 C++ Coding Interview Questions and Answers [2025 Updated]

Last Updated : 17 Jan, 2025

C++ is one of the most popular languages in the software industry for developing software ranging from operating systems, and DBMS to games. That is why it is also popular to be asked to write C++ Programs in live coding sessions in job placement interviews.

This article provides a list of **C++ coding interview questions** for beginners as well as experienced professionals. The questions are designed to test candidates' understanding of the following topics:

- C++ syntax and semantics
- Data structures and algorithms
- Object-oriented programming
- Memory management
- Pointers
- Templates

List of 100 C++ Coding Interview Questions and Answers

Here is a list of 100 C++ coding interview questions and answers

1. Write a C++ Program to Check Whether a Number is a Positive or Negative Number.

```
using namespace std;

int main()
{
    int number;
    number = -100;
    if (number >= 0) {
        cout << number << " is a positive number." << endl;
    }
    else {
        cout << number << " is a negative number." << endl;
    }
    return 0;
}
```

Output

```
-100 is a negative number.
```

2. Write a Program to Find the Greatest of the Three Numbers.

```
// C++ program to find greatest
// among three numbers using
#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 20, c = 30;

    cout << "The Greatest Among Three Numbers is : ";

    if (a >= b && a >= c) {
        cout << a << endl;
    }
    else if (b >= a && b >= c) {
```

```
        cout << c << endl;
    }

    return 0;
}
```

Output

```
The Greatest Among Three Numbers is : 30
```

3. C++ Program To Check Whether Number is Even Or Odd

```
// C++ program to check
// for even or odd
#include <iostream>
using namespace std;

// Returns true if n is
// even, else odd
bool isEven(int n) { return (n % 2 == 0); }

// Driver code
int main()
{
    int n = 247;
    if (isEven(n) == true) {
        cout << "Even" << endl;
    }
    else {
        cout << "Odd";
    }

    return 0;
}
```

For more information, refer to the article – *C++ Program To Check Whether Number is Even Or Odd*

4. Write a Program to Find the ASCII Value of a Character

```cpp
// C++ Program to find ASCII value of a character
#include <iostream>
using namespace std;

int main()
{
    char ch;

    ch = 'A';

    cout << "The ASCII value of " << ch << " is " << int(ch)
         << endl;

    return 0;
}
```

Output

```
The ASCII value of A is 65
```

5. Write a Program to Check Whether a Character is a Vowel or Consonant

```cpp
// C++ Program to print whether a character is vowel
#include <cctype>
#include <iostream>
using namespace std;

int main()
{
```

```
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o'
            || ch == 'u' || ch == 'A' || ch == 'E'
            || ch == 'I' || ch == 'O' || ch == 'U') {
            cout << ch << " is a vowel." << endl;
        }
        else {
            cout << ch << " is a consonant." << endl;
        }
    }
    else {
        cout << ch << " is not an alphabet." << endl;
    }

    return 0;
}
```

Output

```
e is a vowel.
```

6. Write a Program to Print Check Whether a Character is an Alphabet or Not

```
// C++ program to print whether a character is an al
// or not
#include <cctype>
#include <iostream>
using namespace std;

int main()
{
    char ch;

    ch = 'a';

    if (isalpha(ch)) {
```

```
        cout << ch << " is not an alphabet." << endl;
    }

    return 0;
}
```

Output

```
a is an alphabet.
```

7. Write a Program to Find the Length of the String Without using strlen() Function

```
// C++ Program to find the length of a string withou
// strlen()
#include <cstring>
#include <iostream>
using namespace std;

int main()
{
    string str = "GeeksforGeeks";
    int length = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        length++;
    }

    cout << "The length of the string is: " << length
         << endl;

    return 0;
}
```

Output

8. Write a Program to Toggle Each Character in a String

```cpp
// C++ Program to toggle string
#include <cstring>
#include <iostream>
using namespace std;

int main()
{
    string str = "GeeksforGeeks";

    for (int i = 0; str[i] != '\0'; i++) {
        if (islower(str[i])) {
            str[i] = toupper(str[i]);
        }
        else if (isupper(str[i])) {
            str[i] = tolower(str[i]);
        }
    }

    cout << "Toggled string: " << str << endl;

    return 0;
}
```

Output

```
Toggled string: gEEKSFORgEEKS
```

9. Write a Program to Count the Number of Vowels

```cpp
// C++ Program to count the number of vowels
#include <cstring>
#include <iostream>
using namespace std;
```

```
    int vowels = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i'
            || str[i] == 'o' || str[i] == 'u'
            || str[i] == 'A' || str[i] == 'E'
            || str[i] == 'I' || str[i] == 'O'
            || str[i] == 'U') {
            vowels++;
        }
    }

    cout << "Number of vowels in the string: " << vowels
         << endl;

    return 0;
}
```

Output

```
Number of vowels in the string: 9
```

10. Write a Program to Remove the Vowels from a String

```
// C++ Program to remove the vowels from a string
#include <cstring>
#include <iostream>
using namespace std;

int main()
{
    int j = 0;

    string str = "GeeksforGeeks";

    for (int i = 0; str[i] != '\0'; i++) {
```

```
            && str[i] != 'I' && str[i] != 'O'
            && str[i] != 'U') {
            str[j++] = str[i];
        }
    }
    while (j < str.size()) {

        str[j] = '\0';

        j++;
    }
    cout << "String without vowels: " << str << endl;

    return 0;
}
```

Output

```
String without vowels: GksfrGks
```

11. Write a Program to Remove All Characters From a String Except Alphabets

```
// C++ Programto remove all characters from a string
// alphabets
#include <cctype>
#include <iostream>
#include <string>

using namespace std;

string remove_non_alphabets(string str)
{
    string result = "";
    for (char c : str) {
        if (isalpha(c)) {
```

```
    return result;
}

int main()
{
    string str = "Gee$ksfor$geeks";

    cout << "Alphabets only: " << remove_non_alphabets(str)
         << endl;

    return 0;
}
```

Output

```
Alphabets only: Geeksforgeeks
```

12. Write a Program to Remove Spaces From a String

```
// C++ Program to remove spaces from a string
#include <iostream>
#include <string>

using namespace std;

string remove_spaces(string str)
{
    string result = "";
    for (char c : str) {
        if (c != ' ') {
            result += c;
        }
    }
    return result;
}
```

```
    cout << "Without spaces: " << remove_spaces(str)
         << endl;

    return 0;
}
```

Output

```
Without spaces: Gfgtothemoon
```

13. Write a Program to Find the Sum of the First N Natural Numbers

```
// C++ program to find
// Sum of first
// n natural numbers.
#include <iostream>
using namespace std;

// Function to find sum
int findSum(int n)
{
    int sum = 0;
    for (int i = 1; i <= n; i++)
        sum = sum + i;
    return sum;
}

int main()
{
    int n = 7;
    cout << findSum(n);
    return 0;
}
```

```
28
```

14. Write a Program to Find the Factorial of a Number Using Loops

```cpp
// C++ program to find factorial using loops
#include <bits/stdc++.h>
using namespace std;

// function to find factorial
int factorial(int n)
{
    int fact = 1;
    while (n > 1) {
        fact *= n;
        n--;
    }

    return fact;
}

// driver code
int main()
{
    int num = 5;

    cout << factorial(num);

    return 0;
}
```

Output

```
120
```

15. Write a Program to Find a Leap Year or Not

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

bool checkYear(int year)
{
    // leap year
    if (year % 400 == 0)
        return true;

    // Not leap year
    if (year % 100 == 0)
        return false;

    // leap year
    if (year % 4 == 0)
        return true;

    // Not leap year
    return false;
}

int main()
{
    int year = 2000;

    if (checkYear(year))
        cout << "Leap Year";
    else
        cout << "Not a Leap Year";
    return 0;
}
```

Output

```
Leap Year
```

16. Write a Program to Check the Prime Number

```
// Number is prime
#include <iostream>
using namespace std;

bool isPrime(int n)
{
    // base condition
    if (n <= 1)
        return false;

    // Check from 2 to n-1
    for (int i = 2; i < n; i++)
        if (n % i == 0)
            return false;

    return true;
}

int main()
{
    isPrime(21) ? cout << " true\n" : cout << " false\n";
    isPrime(17) ? cout << " true\n" : cout << " false\n";
    return 0;
}
```

Output

```
 false
 true
```

17. Write a Program to Check Palindrome

```
// C++ program to check if a
// number is Palindrome or not
#include <iostream>
using namespace std;
```

```
{
    int ans = 0;
    int temp = n;
    while (temp != 0) {
        ans = (ans * 10) + (temp % 10);
        temp = temp / 10;
    }

    return (ans == n);
}

int main()
{
    int n = 12321;

    if (checkPalindrome(n) == 1) {
        cout << "Yes\n";
    }
    else {
        cout << "No\n";
    }

    return 0;
}
```

Output

```
Yes
```

18. Write a Program to Check Whether a Number is an Armstrong Number or Not

```
// C++ Program to check
// if number is Armstrong
// or not
#include <iostream>
```

```
{
    int n = 153;
    int temp = n;
    int ans = 0;

    // function to calculate
    // the sum of individual digits
    while (n > 0) {

        int rem = n % 10;
        ans = (ans) + (rem * rem * rem);
        n = n / 10;
    }

    // condition to check
    if (temp == ans) {
        cout << ("Yes, it is Armstrong Number");
    }
    else {
        cout << ("No, it is not an Armstrong Number");
    }
    return 0;
}
```

Output

```
Yes, it is Armstrong Number
```

19. Write a Program to Find the Nth Term of the Fibonacci Series

```
// C++ Program to Find the
// Nth Term of the Fibonacci Series
#include <iostream>
using namespace std;

int fib(int n)
```

```
        return first;

    for (int i = 2; i <= n; i++) {
        ans = first + second;
        first = second;
        second = ans;
    }

    return ans;
}

int main()
{
    int n = 13;

    cout << fib(n);
    return 0;
}
```

Output

```
233
```

20. Write a Program to Calculate the Greatest Common Divisor of Two Numbers

```
// C++ program to find
// GCD of two numbers
#include <iostream>

using namespace std;

// Function to return gcd of a and b
int gcd(int a, int b)
{
    int result = min(a, b);
```

```
            break;
        }
        result--;
    }

    return result;
}

int main()
{
    int a = 54, b = 33;

    cout << "GCD: " << gcd(a, b);

    return 0;
}
```

Output

```
GCD: 3
```

21. Write a Program to Calculate the Lowest Common Multiple (LCM) of Two Numbers

```
// C++ program to
// Find LCM of two numbers
#include <iostream>
using namespace std;

long long gcd(long long int a, long long int b)
{
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

```
    long long result = (a / gcd(a, b)) * b;
    return result;
}

int main()
{
    int a = 24, b = 13;
    cout << "LCM : " << lcm(a, b);
    return 0;
}
```

Output

```
LCM : 312
```

22. Write a Program for Finding the Roots of a Quadratic Equation

```
// C++ program to find
// Roots of a quadratic equation
#include <iostream>
#include <math.h>
using namespace std;

// Prints roots of quadratic equation ax*2 + bx + c
void findRoots(int a, int b, int c)
{
    // If a is 0, then equation is not quadratic
    if (a == 0) {
        cout << "Invalid";
        return;
    }

    // Formulae to calculate D
    int d = b * b - 4 * a * c;

    // Formulae to calculate
```

```
    // Conditons for checking root
    if (d > 0) {
        cout << "Roots are real and different \n";
        cout << (double)(-b + sqrt_val) / (2 * a) << "\n"
             << (double)(-b - sqrt_val) / (2 * a);
    }
    else if (d == 0) {
        cout << "Roots are real and same \n";
        cout << -(double)b / (2 * a);
    }
    else {
        cout << "Roots are complex \n";
        cout << -(double)b / (2 * a) << " + i"
             << sqrt_val / (2 * a) << "\n"
             << -(double)b / (2 * a) << " - i"
             << sqrt_val / (2 * a);
    }
}

int main()
{
    int a = 1, b = 4, c = 4;

    findRoots(a, b, c);

    return 0;
}
```

Output

```
Roots are real and same
-2
```

23. Write a Program to Find the Smallest and Largest Element in an Array

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

// Function to find the minimum
// and maximum of the array
void findMinMax(int arr[], int n)
{
    int mini = arr[0];
    int maxi = arr[0];

    for (int i = 0; i < n; i++) {
        if (arr[i] < mini) {
            mini = arr[i];
        }
        else if (arr[i] > maxi) {
            maxi = arr[i];
        }
    }

    cout << "Min: " << mini << endl;
    cout << "Max: " << maxi << endl;
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };
    int N = sizeof(arr) / sizeof(arr[0]);

    findMinMax(arr, N);

    return 0;
}
```

Output

```
Min: 1
Max: 5
```

24. Write a Program to Find the Second Smallest Element in an Array

```cpp
// C++ program to find
// Second smallest elements
#include <climits>
#include <iostream>
using namespace std;

void print2Smallest(int arr[], int n)
{
    int first, second;

    if (n < 2) {
        cout << " Invalid Input ";
        return;
    }

    first = second = INT_MAX;
    for (int i = 0; i < n; i++) {
        // If current element is smaller than first
        // Then update both first and second
        if (arr[i] < first) {
            second = first;
            first = arr[i];
        }

        // If arr[i] is in between first and second
        // Then update second
        else if (arr[i] < second && arr[i] != first)
            second = arr[i];
    }
    if (second == INT_MAX)
        cout << "There is no second smallest element\n";
    else
        cout << " Second smallest element is " << second
             << endl;
```

```
{
    int arr[] = { 21, 3, 15, 41, 34, 10 };
    int n = sizeof(arr) / sizeof(arr[0]);

    print2Smallest(arr, n);

    return 0;
}
```

Output

```
Second smallest element is 10
```

25. Write a Program to Calculate the Sum of Elements in an Array

```
// C++ Program to calculate
// sum of elements in an array
#include <iostream>
using namespace std;

int sum(int arr[], int n)
{
    int sum = 0;

    for (int i = 0; i < n; i++)
        sum += arr[i];

    return sum;
}

int main()
{
    int arr[] = { 1, 23, 54, 12, 9 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Sum: " << sum(arr, n);
```

```
}
```

Output

```
Sum: 99
```

26. Write a Program to Check if the Given String is Palindrome or Not

```
// C++ program for checking
// if it is Palindrome or not
#include <iostream>
using namespace std;

string isPalindrome(string S)
{
    for (int i = 0; i < S.length() / 2; i++) {
        if (S[i] != S[S.length() - i - 1]) {
            return "No";
        }
    }

    return "Yes";
}

int main()
{
    string S = "GeekeeG";

    cout << isPalindrome(S);

    return 0;
}
```

Output

27. Write a Program to Check if Two Strings are Anagram or Not

```cpp
// C++ program to check if two strings
// Are anagrams of each other
#include <iostream>
using namespace std;

#define NO_OF_CHARS 256

bool areAnagram(char* str1, char* str2)
{
    // Create 2 count arrays and initialize all values as 0
    int count1[NO_OF_CHARS] = { 0 };
    int count2[NO_OF_CHARS] = { 0 };
    int i;

    // For each character in input strings, increment count
    // in the corresponding count array
    for (i = 0; str1[i] && str2[i]; i++) {
        count1[str1[i]]++;
        count2[str2[i]]++;
    }

    if (str1[i] || str2[i])
        return false;

    // Compare count arrays
    for (i = 0; i < NO_OF_CHARS; i++)
        if (count1[i] != count2[i])
            return false;

    return true;
}

int main()
{
    char str1[] = "Geek";
```

```
        cout << "The two strings are anagram of each other";
    else
        cout << "The two strings are not anagram of each "
                "other";

    return 0;
}
```

Output

```
The two strings are not anagram of each other
```

28. Write a Program to Print a Diamond Pattern

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```
// C++ program to print
// Diamond shape
#include <iostream>
using namespace std;

void printDiamond(int n)
{
    int space = n - 1;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < space; j++)
            cout << " ";
```

```
        cout << endl;
        space--;
    }

    space = 0;

    // run loop (parent loop)
    for (int i = n; i > 0; i--) {
        for (int j = 0; j < space; j++)
            cout << " ";

        // Print i stars
        for (int j = 0; j < i; j++)
            cout << "* ";

        cout << endl;
        space++;
    }
}

int main()
{
    printDiamond(5);
    return 0;
}
```

Output

```
    *
   * *
  * * *
 * * * *
* * * * *
* * * * *
 * * * *
  * * *
   * *
```

29. Write a Program to Print a Pyramid Pattern

```
    *
   ***
  *****
 *******
```

```cpp
// C++ Program to
// Print Pyramid pattern
#include <iostream>
using namespace std;

void pattern(int n)
{
    int k = 2 * n - 2;

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

        for (int j = 0; j < k; j++)
            cout << " ";

        k = k - 1;
        for (int j = 0; j <= i; j++) {
            // Printing stars
            cout << "* ";
        }
        cout << endl;
    }
}

int main()
{
    int n = 5;

    pattern(n);
    return 0;
}
```

```
        *
       * *
      * * *
     * * * *
    * * * * *
```

30. Write a Program to Print the Hourglass Pattern

```
* * * * * * * * * *
  * * * * * * * *
    * * * * * *
      * * * *
        * *
      * * * *
    * * * * * *
  * * * * * * * *
* * * * * * * * * *
```

```
// C Program to print hourglass pattern
#include <iostream>
using namespace std;

// function to print hourglass pattern
void hourglass(int rows)
{

    // first outer loop to iterate each row
    for (int i = 0; i < 2 * rows - 1; i++) {

        // assigning comparator
        int comp;
        if (i < rows) {
            comp = 2 * i + 1;
        }
        else {
```

```
        // first inner loop to print leading spaces
        for (int j = 0; j < comp; j++) {
            cout << ' ';
        }

        // second inner loop to print star *
        for (int k = 0; k < 2 * rows - comp; k++) {
            cout << "* ";
        }
        cout << '\n';
    }
}

int main()
{
    hourglass(5);
    return 0;
}
```

Output

```
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

31. Write a Program to Print the Rotated Hourglass Pattern

```
*                       *
* *                   * *
```

```
* * * * * * *       * * * * * * *
* * * * * * * * * * * * * * * * *
* * * * * * *       * * * * * * *
* * * * * *           * * * * * *
* * * * *               * * * * *
* * * *                   * * * *
* * *                       * * *
* *                           * *
*                               *
```

```
// C++ Program to print
// star pattern given
#include <iostream>
using namespace std;

void pattern(int n)
{
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= i; j++) {
            cout << "* ";
        }

        int spaces = 2 * (n - i);
        for (int j = 0; j < spaces; j++) {
            cout << "  ";
        }

        for (int j = 0; j <= i; j++) {
            cout << "* ";
        }

        cout << endl;
    }

    // Printing bottom part.
    for (int i = n - 1; i >= 0; i--) {

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

```
        int spaces = 2 * (n - i);
        for (int j = 0; j < spaces; j++) {
            cout << "  ";
        }

        for (int j = 0; j <= i; j++) {
            cout << "* ";
        }

        cout << endl;
    }
}

int main()
{
    int n = 5;

    pattern(n);

    return 0;
}
```

Output

```
*                       *
* *                   * *
* * *               * * *
* * * *           * * * *
* * * * *       * * * * *
* * * * * * * * * * * * *
* * * * *       * * * * *
* * * *           * * * *
* * *               * * *
* *                   * *
*                       *
```

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

int main()
{
    int rows = 5;

    for (int i = 1; i <= rows; i++) {
        for (int j = rows; j >= i; j--) {
            cout << " ";
        }
        for (int k = 1; k <= (2 * i - 1); k++) {
            cout << "*";
        }
        cout << endl;
    }

    return 0;
}
```

Output

```
    *
   ***
  *****
 *******
*********
```

33. Write a Program to print an Inverted Pyramid

```cpp
// C++ Program to print inverted pyramid
#include <iostream>
using namespace std;

int main()
{
```

```
        for (int j = rows; j > i; j--) {
            cout << " ";
        }
        for (int k = 1; k <= (2 * i - 1); k++) {
            cout << "*";
        }
        cout << endl;
    }

    return 0;
}
```

Output

```
*********
 *******
  *****
   ***
    *
```

34. Write a Program to Print a Triangle Star Pattern

```
// C++ Program to print a triangle star patter
#include <iostream>
using namespace std;

int main()
{
    int rows;

    rows = 5;

    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            cout << "*";
        }
```

```
    return 0;
}
```

Output

```
*
**
***
****
*****
```

35. Write a Program to Print Floyd's Triangle

```
1
2 3
4 5 6
7 8 9 10
```

```
// C Program to print the Floyd's Triangle
#include <stdio.h>

int main()
{
    int rows = 4;
    int n = 1;

    // outer loop to print all rows
    for (int i = 0; i < rows; i++) {

        // innter loop to print abphabet in each row
        for (int j = 0; j <= i; j++) {
            printf("%d ", n++);
        }
        printf("\n");
    }
```

Output

```
1
2 3
4 5 6
7 8 9 10
```

36. Write a Program to Print the Pascal Triangle

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5   10  10  5   1
```

```cpp
// C++ program to print
// Pascal’s Triangle
#include <iostream>
using namespace std;

void printPascal(int n)
{

    int arr[n][n];

    for (int line = 0; line < n; line++) {
        // Every line has number of integers
        // equal to line number
        for (int i = 0; i <= line; i++) {
            // First and last values in every row are 1
            if (line == i || i == 0)
                arr[line][i] = 1;
            else
```

```
            cout << "\n";
        }
}

int main()
{
    int n = 6;
    printPascal(n);
    return 0;
}
```

Output

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

37. Write a Program to Print the Given String in Reverse Order

```
// C++ Program to reversea string
#include <cstring>
#include <iostream>
using namespace std;

int main()
{

    int len;

    string str = "GeeksforGeeks";

    len = str.size();
```

```
    }
    cout << endl;

    return 0;
}
```

Output

```
Reverse of the string: skeeGrofskeeG
```

38. Write a C++ Program to Print the Given String in Reverse Order Using Recursion

```
// C++ Program to
// Reverse string using
// recursion
#include <iostream>
using namespace std;

void reverse_str(string& s, int n, int i)
{
    if (n <= i) {
        return;
    }

    swap(s[i], s[n]);
    reverse_str(s, n - 1, i + 1);
}

int main()
{
    string str = "GeeksforGeeks";
    reverse_str(str, str.length() - 1, 0);
    cout << str << endl;
}
```

```
skeeGrofskeeG
```

39. Write a Program to Check if the Given String is Palindrome or not Using Recursion

```cpp
// C++ program to check
// Whether a given number
// Is palindrome or not
#include <bits/stdc++.h>
using namespace std;

bool isPalRec(char str[], int s, int n)
{
    // If there is only one character
    if (s == n)
        return true;

    // If first and last
    // characters do not match
    if (str[s] != str[n])
        return false;

    if (s < n + 1)
        return isPalRec(str, s + 1, n - 1);

    return true;
}

bool isPalindrome(char str[])
{
    int n = strlen(str);

    if (n == 0)
        return true;

    return isPalRec(str, 0, n - 1);
```

```
{
    char str[] = "GeeKeeG";

    if (isPalindrome(str))
        cout << "Yes";
    else
        cout << "No";

    return 0;
}
```

Output

```
Yes
```

40. Write a Program to Calculate the Length of the String Using Recursion

```
// C++ Program for calculating
// the length of string
#include <iostream>
using namespace std;

int cal(char* str)
{
    // base condition
    if (*str == '\0')
        return 0;
    else
        return 1 + cal(str + 1);
}

int main()
{
    char str[] = "GeeksforGeeks";
    cout << cal(str);
```

Output

```
13
```

41. Write a Program to Calculate the Factorial of a Number Using Recursion

```cpp
// C++ program to calculate
// Factorial of given number
#include <iostream>
using namespace std;

unsigned long long factorial(unsigned long long n)
{
    if (n == 0 || n == 1)
        return 1;
    return n * factorial(n - 1);
}

int main()
{
    unsigned long long num = 15;

    cout << "Factorial of " << num << " is "
         << factorial(num) << endl;

    return 0;
}
```

Output

```
Factorial of 15 is 1307674368000
```

42. Write a Program to Count the Sum of Numbers in a String

```cpp
#include <sstream>
#include <string>

using namespace std;

int sum_of_numbers(string str)
{
    int sum = 0;
    for (char ch : str) {
        if (isdigit(ch)) {
            sum += ch - '0';
        }
    }
    return sum;
}

int main()
{
    string str;

    str = "1234";

    cout << "Sum of numbers: " << sum_of_numbers(str)
         << endl;

    return 0;
}
```

Output

```
Sum of numbers: 10
```

43. Write a Program to Print All Natural Numbers up to N Without Using a Semi-Colon

```cpp
// C++ program to print all natural numbers upto
```

```
using namespace std;
#define N 10

int main()
{
    static int x = 1;
    if (cout << x << " " && x++ < N && main()) {
    }
    return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

44. Write a Program to Swap the Values of Two Variables Without Using any Extra Variable

```
// C++ program to check
// If two numbers are equal
#include <iostream>
using namespace std;

int main()
{
    int x = 3;
    int y = 4;

    cout << "X : " << x << endl;
    cout << "Y : " << y << endl;

    x = x + y;
    y = x - y;
    x = x - y;

    cout << endl;
```

```
    cout << "Y : " << y << endl;

    return 0;
}
```

Output

```
X : 3
Y : 4

After:
X : 4
Y : 3
```

45. Write a Program to Print the Maximum Value of an Unsigned int Using One's Complement (~) Operator

```
// C++ program to print maximum value of
// unsigned int.
#include <iostream>

using namespace std;

int main()
{
    unsigned int max;
    max = 0;
    max = ~max;

    cout << "Max value possible : " << max;

    return 0;
}
```

Output

46. Write a Program to Check for the Equality of Two Numbers Without Using Arithmetic or Comparison Operator

```cpp
// C++ Program to equality of
// Two numbers without using
// Arithmetic or comparison operator
#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 10;

    if (a ^ b)
        cout << "Not-Equal";
    else
        cout << "Equal";

    return 0;
}
```

Output

```
Equal
```

47. Write a Program to Find the Maximum and Minimum of the Two Numbers Without Using the Comparison Operator

```cpp
// C++ program to find
// maximum and minimum of
// Two numbers without using
// loop and conditions
#include <iostream>

using namespace std;
```

```
    int a = 5, b = 10;

    cout << "max :" << (((a + b) + abs(a - b)) / 2) << endl;
    cout << "min :" << (((a + b) - abs(a - b)) / 2) << endl;

    return 0;
}
```

Output

```
max :10
min :5
```

48. Write a Program for Octal to Decimal Conversion

```
// C++ Program to convert ocatal to decimal
#include <cmath>
#include <iostream>

using namespace std;

int main()
{
    int oct, dec = 0, place = 0;
    // 67 is an octal number with binary equivalent 110000
    oct = 67;

    int temp = oct;
    while (temp) {
        int lastDigit = temp % 10;
        temp /= 10;
        dec += lastDigit * pow(8, place);
        ++place;
    }

    cout << "Decimal equivalent is: " << dec << endl;
```

```
}
```

Output

```
Decimal equivalent is: 55
```

49. Write a Program for Hexadecimal to Decimal Conversion

```cpp
// C++ Program to convert hexadecimal to decimal con
#include <cmath>
#include <iostream>

using namespace std;

int hexToDecimal(char hexDigit)
{
    if (hexDigit >= '0' && hexDigit <= '9') {
        return int(hexDigit - '0');
    }
    else if (hexDigit >= 'A' && hexDigit <= 'F') {
        return int(hexDigit - 'A' + 10);
    }
    else if (hexDigit >= 'a' && hexDigit <= 'f') {
        return int(hexDigit - 'a' + 10);
    }
    return -1;
}

int main()
{
    string hex;
    int decimal = 0, place = 0;

    hex = "67";

    int n = hex.length();
```

```
        place++;
    }

    cout << "Decimal equivalent " << decimal << endl;

    return 0;
}
```

Output

```
Decimal equivalent 103
```

50. Write a Program for Decimal to Binary Conversion

```
// c++ program to convert decimal to binary
#include <bitset>
#include <iostream>

using namespace std;

int main()
{
    int decimal = 7;

    // simplest method to convert decimal to binary
    bitset<32> binary(decimal);

    cout << "Binary equivalent: " << binary << endl;

    return 0;
}
```

Output

```
Binary equivalent: 00000000000000000000000000000111
```

```cpp
// C++ Program to convert decimal to octal equivalen
#include <cmath>
#include <iostream>

using namespace std;

int main()
{
    int decimal, octal = 0, place = 1;

    decimal = 55;

    int temp = decimal;
    while (temp) {
        int lastDigit = temp % 8;
        temp /= 8;
        octal += lastDigit * place;
        place *= 10;
    }

    cout << "Octal equivalent " << octal << endl;

    return 0;
}
```

Output

```
Octal equivalent 67
```

52. Write a Program for Decimal to Hexadecimal Conversion

```cpp
// C++ program to convert decimal to hexadecimal
#include <cmath>
#include <iostream>
#include <string>
```

```
    string hexadecimal = "";
    char hexaDecimals[16]
        = { '0', '1', '2', '3', '4', '5', '6', '7',
            '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
    while (decimal > 0) {
        int remainder = decimal % 16;
        hexadecimal = hexaDecimals[remainder] + hexadecimal;
        decimal /= 16;
    }
    return hexadecimal;
}

int main()
{
    int decimal = 103;

    cout << "Hexadecimal equivalent: "
         << decimalToHexa(decimal) << endl;

    return 0;
}
```

Output

```
Hexadecimal equivalent: 67
```

53. Write a Program for Binary to Octal Conversion

```
// C++ implementation to convert a binary number
// to octal number
#include <bits/stdc++.h>
using namespace std;

// function to create map between binary
// number and its equivalent octal
void createMap(unordered_map<string, char>* um)
```

```
    (*um)["010"] = '2';
    (*um)["011"] = '3';
    (*um)["100"] = '4';
    (*um)["101"] = '5';
    (*um)["110"] = '6';
    (*um)["111"] = '7';
}

// Function to find octal equivalent of binary
string convertBinToOct(string bin)
{
    int l = bin.size();
    int t = bin.find_first_of('.');

    // length of string before '.'
    int len_left = t != -1 ? t : l;

    // add min 0's in the beginning to make
    // left substring length divisible by 3
    for (int i = 1; i <= (3 - len_left % 3) % 3; i++)
        bin = '0' + bin;

    // if decimal point exists
    if (t != -1) {
        // length of string after '.'
        int len_right = l - len_left - 1;

        // add min 0's in the end to make right
        // substring length divisible by 3
        for (int i = 1; i <= (3 - len_right % 3) % 3; i++)
            bin = bin + '0';
    }

    // create map between binary and its
    // equivalent octal code
    unordered_map<string, char> bin_oct_map;
    createMap(&bin_oct_map);
```

```
    while (1) {
        // one by one extract from left, substring
        // of size 3 and add its octal code
        octal += bin_oct_map[bin.substr(i, 3)];
        i += 3;
        if (i == bin.size())
            break;

        // if '.' is encountered add it to result
        if (bin.at(i) == '.') {
            octal += '.';
            i++;
        }
    }

    // required octal number
    return octal;
}

// Driver program to test above
int main()
{
    string bin = "1111001010010100001.010110110011011";
    cout << "Octal number = " << convertBinToOct(bin);
    return 0;
}
```

Output

```
Octal number = 1712241.26633
```

54. Write a Program for Octal to Binary Conversion

```
// C++ program to convert
// Octal number to Binary
```

```
// Function to convert an
// Octal to Binary Number
string OctToBin(string octnum)
{
    long int i = 0;

    string binary = "";

    while (octnum[i]) {
        switch (octnum[i]) {
        case '0':
            binary += "000";
            break;
        case '1':
            binary += "001";
            break;
        case '2':
            binary += "010";
            break;
        case '3':
            binary += "011";
            break;
        case '4':
            binary += "100";
            break;
        case '5':
            binary += "101";
            break;
        case '6':
            binary += "110";
            break;
        case '7':
            binary += "111";
            break;
        default:
            cout << "\nInvalid Octal Digit " << octnum[i];
            break;
        }
```

```
    return binary;
}

// Driver code
int main()
{
    // Get the Hexadecimal number
    string octnum = "345";

    // Convert Octal to Binary
    cout << "Equivalent Binary Value = "
         << OctToBin(octnum);

    return 0;
}
```

Output

```
Equivalent Binary Value = 011100101
```

55. Write a Program to Implement the Use of Encapsulation

```
// C++ Program to implement
// The concept of Encapsulation
#include <iostream>
using namespace std;

class Encapsulation {
private:
    // data hidden from outer functions
    int x;

public:
    // function to set value of
    // variable x
    void setter(int a) { x = a; }
```

```
    int getter() { return x; }
};

int main()
{
    Encapsulation obj;

    obj.setter(13);

    cout << obj.getter();

    return 0;
}
```

Output

```
13
```

56. Write a Program to Implement the Concept of Abstraction

```
// C++ Program to implement
// Working of Abstraction
#include <iostream>
using namespace std;

class implementAbstraction {
private:
    int p, q;

public:
    // method to set values of
    // private members
    void setter(int x, int y)
    {
        p = x;
        q = y;
```

```
    {
        cout << "p = " << p << endl;
        cout << "q = " << q << endl;
    }
};

int main()
{
    implementAbstraction obj;

    obj.setter(1, 2);
    obj.display();

    return 0;
}
```

Output

```
p = 1
q = 2
```

57. Write a Program to Implement the Concept of Compile-Time Polymorphism or Function Overloading

```
// C++ program to demonstrate
// Function overloading or
// Compile-time Polymorphism
#include <iostream>

using namespace std;

class Geeks {
public:
    // Function same name different
    // Parameters
    void func(int x)
```

```
    void func(double x)
    {
        cout << "value of x is " << x << endl;
    }

    void func(int x, int y)
    {
        cout << "value of x and y is " << x << ", " << y
             << endl;
    }
};

int main()
{
    Geeks obj1;

    // Function being called depends
    // on the parameters passed
    // func() is called with int value
    obj1.func(10);

    // func() is called with double value
    obj1.func(5.321);

    // func() is called with 2 int values
    obj1.func(94, 32);
    return 0;
}
```

Output

```
value of x is 10
value of x is 5.321
value of x and y is 94, 32
```

58. Write a Program to Implement the Concept of Operator

```cpp
// C++ program to demonstrate
// Operator Overloading
#include <iostream>
using namespace std;

class Complex {
private:
    int real, imag;

public:
    Complex(int r = 0, int i = 0)
    {
        real = r;
        imag = i;
    }

    // This is automatically called
    // when '+' is used
    Complex operator+(Complex const& obj)
    {
        Complex res;
        res.real = real + obj.real;
        res.imag = imag + obj.imag;
        return res;
    }
    void print() { cout << real << " + " << imag << "i\n"; }
};

int main()
{
    Complex c1(15, 5), c2(3, 5);

    Complex c3 = c1 + c2;
    c3.print();
}
```

Output

59. Write a Program to Implement the Concept of Function Overriding or Runtime Polymorphism

```cpp
// C++ program for implementation
// of Function Overloading or
// Compile time Polymorphism
#include <iostream>
using namespace std;

class base {
public:
    virtual void print()
    {
        cout << "print base class" << endl;
    }

    void show() { cout << "show base class" << endl; }
};

class derived : public base {
public:
    void print() { cout << "print derived class" << endl; }

    void show() { cout << "show derived class" << endl; }
};

int main()
{
    base* bptr;
    derived d;
    bptr = &d;

    bptr->print();

    // Non-virtual function, binded
    // at compile time
    bptr->show();
```

```
}
```

Output

```
print derived class
show base class
```

60. Write a Program to Implement Single-Level Inheritance

```
// C++ Program to implement
// Single level inheritance
#include <iostream>
#include <string.h>

using namespace std;

class Person {
    int id;
    char name[100];

public:
    void set_p(int id, char* name)
    {
        strcpy(this->name, name);
        this->id = id;
    }

    void display_p()
    {
        cout << endl << id << "\t" << name << "\t";
    }
};

class Student : private Person {
    char course[50];
    int fee;
```

```
    {
        set_p(id, name);

        strcpy(this->course, course);

        this->fee = fee;
    }

    void display_s()
    {
        display_p();

        cout << course << "\t" << fee << endl;
    }
};

main()
{
    Student s;
    char name[] = "XYZ";
    char course[] = "ABC";
    s.set_s(132451, name, course, 100000);
    s.display_s();
    return 0;
}
```

Output

```
132451    XYZ    ABC    100000
```

61. Write a Program to Create a Class for Complex Numbers

```
// C++ Program to create a class of complex numbers
#include <bits/stdc++.h>
using namespace std;
```

```
    double img;
};

// complex clss
class Complex {
private:
    struct c num;

public:
    // constructors
    Complex() {}
    Complex(double real, double img)
    {
        num.img = img;
        num.real = real;
    }
    Complex(Complex& var)
    {
        num.img = var.num.img;
        num.real = var.num.real;
    }

    // utility functions
    void print()
    {
        cout << num.real << " + i" << num.img << endl;
    }

    double imag() { return num.img; }
    double real() { return num.real; }

    // overloaded operators
    Complex operator+(Complex& obj1)
    {
        Complex var;
        var.num.real = num.real + obj1.num.real;
        var.num.img = num.img + obj1.num.img;
```

```
    {
        Complex var;
        var.num.real = num.real - obj1.num.real;
        var.num.img = num.img - obj1.num.img;

        return var;
    }

    Complex operator*(Complex& obj1)
    {
        Complex var;
        var.num.real = num.real * obj1.num.real
                       - num.img * obj1.num.img;
        var.num.img = num.real * obj1.num.img
                      + num.img * obj1.num.real;

        return var;
    }
};

// driver code
int main()
{
    Complex a(11, 12), b(5, 8);
    Complex c;
    c = a + b;

    a.print();
    b.print();
    c.print();

    return 0;
}
```

Output

```
11 + i12
```

62. Write a Program to Implement the Inch Feet System

```cpp
// C++ Program to create a class of inchFeet length
#include <bits/stdc++.h>
using namespace std;

// inch-feet length system datatype
struct c {
    double feet;
    double inch;
};

// inchFeet class
class inchFeet {
private:
    struct c length;

public:
    // constructors
    inchFeet() {}
    inchFeet(double feet, double inch)
    {
        length.inch = inch;
        length.feet = feet;
    }
    inchFeet(inchFeet& var)
    {
        length.inch = var.length.inch;
        length.feet = var.length.feet;
    }

    // utility functions
    void print()
    {
        cout << length.feet << " feet and " << length.inch
             << " inches" << endl;
    }
```

```
    // overloaded operators
    inchFeet operator+(inchFeet& obj1)
    {
        inchFeet var;
        var.length.feet = length.feet + obj1.length.feet;
        var.length.inch = length.inch + obj1.length.inch;
        if (var.length.inch >= 12.0) {
            var.length.feet++;
            var.length.inch - 12.0;
        }

        return var;
    }
    inchFeet operator-(inchFeet& obj1)
    {
        inchFeet var;
        struct c temp = length;
        if (temp.feet > obj1.length.feet) {
            if (temp.inch < obj1.length.inch) {
                temp.feet--;
                temp.inch += 12;
            }
            var.length.feet = temp.feet - obj1.length.feet;
            var.length.inch = temp.inch - obj1.length.inch;
        }
        else {
            cout << "Negative Length is not Possible\n";
        }
        return var;
    }
};

// driver code
int main()
{
    inchFeet a(11, 4), b(5, 8);
    inchFeet c;
```

```
    b.print();
    c.print();

    return 0;
}
```

Output

```
11 feet and 4 inches
5 feet and 8 inches
5 feet and 8 inches
```

63. Write a Program to Implement Bubble Sort

```
// C++ program to implement
// of Bubble sort
#include <iostream>
using namespace std;

// Function to sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already
        // in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
}

// Function to print an array
void printArray(int arr[], int size)
{
    int i;
```

```
}

int main()
{
    int arr[] = { 3, 1, 4, 2, 5 };
    int N = sizeof(arr) / sizeof(arr[0]);

    bubbleSort(arr, N);

    cout << "Sorted array: ";
    printArray(arr, N);
    return 0;
}
```

Output

```
Sorted array: 1 2 3 4 5
```

64. Write a Program to Implement Insertion Sort

```
// C++ program to implement
// Insertion sort
#include <bits/stdc++.h>
using namespace std;

// Function to sort using
// Insertion
void insertion_sort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
```

```
    }
}

// Print array
void print_array(int arr[], int n)
{
    cout << " Sorted array:";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main()
{
    int arr[] = { 1, 4, 3, 2, 5 };
    int N = sizeof(arr) / sizeof(arr[0]);

    insertion_sort(arr, N);
    print_array(arr, N);

    return 0;
}
```

Output

```
 Sorted array:1 2 3 4 5
```

65. Write a Program to Implement Selection Sort

```
// C++ program to implement
// Selection sort
#include <iostream>
using namespace std;

// Swap function
void swap(int* p, int* q)
```

```
    *q = temp;
}

void selectionSort(int arr[], int n)
{
    int min_index;

    for (int i = 0; i < n - 1; i++) {
        min_index = i;
        for (int j = i + 1; j < n; j++)
            if (arr[j] < arr[min_index])
                min_index = j;

        // Swap the found minimum element
        // with the first element
        if (min_index != i)
            swap(&arr[min_index], &arr[i]);
    }
}

// Print Array
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main()
{
    int arr[] = { 5, 4, 3, 2, 1 };
    int n = sizeof(arr) / sizeof(arr[0]);

    selectionSort(arr, n);

    cout << "Sorted array: ";
    printArray(arr, n);
```

Output

```
Sorted array: 1 2 3 4 5
```

66. Write a Program to Implement Merge Sort

```cpp
// C++ program to implement
// Merge Sort
#include <iostream>
using namespace std;

// Merge Sorted arrays
void merge(int array[], int const left, int const mid,
           int const right)
{
    auto const subArrayOne = mid - left + 1;
    auto const subArrayTwo = right - mid;

    // Create temp arrays
    auto *leftArray = new int[subArrayOne],
         *rightArray = new int[subArrayTwo];

    // Copy data to temp arrays leftArray[] and rightArray[]
    for (auto i = 0; i < subArrayOne; i++)
        leftArray[i] = array[left + i];
    for (auto j = 0; j < subArrayTwo; j++)
        rightArray[j] = array[mid + 1 + j];

    auto indexOfSubArrayOne = 0, indexOfSubArrayTwo = 0;
    int indexOfMergedArray = left;

    // Merge the temp arrays back into array[left..right]
    while (indexOfSubArrayOne < subArrayOne
           && indexOfSubArrayTwo < subArrayTwo) {
        if (leftArray[indexOfSubArrayOne]
```

```
            indexOfSubArrayOne++;
        }
        else {
            array[indexOfMergedArray]
                = rightArray[indexOfSubArrayTwo];
            indexOfSubArrayTwo++;
        }
        indexOfMergedArray++;
    }

    // Copying remaing elements
    while (indexOfSubArrayOne < subArrayOne) {
        array[indexOfMergedArray]
            = leftArray[indexOfSubArrayOne];
        indexOfSubArrayOne++;
        indexOfMergedArray++;
    }

    while (indexOfSubArrayTwo < subArrayTwo) {
        array[indexOfMergedArray]
            = rightArray[indexOfSubArrayTwo];
        indexOfSubArrayTwo++;
        indexOfMergedArray++;
    }
    delete[] leftArray;
    delete[] rightArray;
}

void mergeSort(int array[], int const begin, int const end)
{
    // base condition
    if (begin >= end)
        return;

    auto mid = begin + (end - begin) / 2;
    mergeSort(array, begin, mid);
    mergeSort(array, mid + 1, end);
    merge(array, begin, mid, end);
```

```
void print_array(int A[], int size)
{
    for (auto i = 0; i < size; i++)
        cout << A[i] << " ";
}

int main()
{
    int arr[] = { 5, 6, 3, 10, 1, 4, 9 };
    auto arr_size = sizeof(arr) / sizeof(arr[0]);

    cout << "Array: ";
    print_array(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    cout << "\nSorted array: ";
    print_array(arr, arr_size);
    return 0;
}
```

Output

```
Array: 5 6 3 10 1 4 9
Sorted array: 1 3 4 5 6 9 10
```

67. Write a Program to Implement Quick Sort

```
// C++ Program to implement
// QuickSort
#include <iostream>
using namespace std;

// Swap  elements
void swap(int* a, int* b)
{
```

```
}

// Partition function to check pivot location
int partition(int arr[], int low, int high)
{
    int pivot = arr[high]; // pivot
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller than the pivot
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

// Quick Sort function
void quickSort(int arr[], int low, int high)
{
    if (low < high) {
        // pi is partitioning index, arr[p] is now
        // at right place
        int pi = partition(arr, low, high);

        // Separately sort elements before
        // partition and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

//  Print  Array
void printArray(int arr[], int size)
{
    int i;
```

```
}

int main()
{
    int arr[] = { 2, 5, 6, 9, 1, 3, 4 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Array: ";
    printArray(arr, n);

    quickSort(arr, 0, n - 1);

    cout << "Sorted array: ";
    printArray(arr, n);
    return 0;
}
```

Output

```
Array: 2 5 6 9 1 3 4
Sorted array: 1 2 3 4 5 6 9
```

68. Write a Program to Implement Linear Search

```
// C++ Program to implement
// Linear Sort
#include <iostream>
using namespace std;

int search(int arr[], int N, int x)
{
    int i;
    for (i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
```

```
{
    int arr[] = { 5, 4, 1, 6, 10, 9, 23, 2 };
    int x = 9;
    int N = sizeof(arr) / sizeof(arr[0]);

    int result = search(arr, N, x);

    if (result == -1)
        cout << "Element is not present in array";
    else
        cout << "Element is present at index " << result;
    return 0;
}
```

Output

```
Element is present at index 5
```

69. Write a Program to Implement Binary Search

```
// C++ program to implement
// Binary Search
#include <iostream>
using namespace std;

// Binary Search Function
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        // Middle element
        int mid = l + (r - l) / 2;

        if (arr[mid] == x)
            return mid;

        if (arr[mid] > x)
```

```
    }

    // We reach here when element is not
    // present in array
    return -1;
}

int main(void)
{
    int arr[] = { 1, 2, 3, 4, 5, 6 };
    int x = 5;
    int n = sizeof(arr) / sizeof(arr[0]);

    int result = binarySearch(arr, 0, n - 1, x);

    if (result == -1)
        cout << "Element is not present in array";
    else
        cout << "Element is present at index " << result;
    return 0;
}
```

Output

```
Element is present at index 4
```

70. Write a Program to Find the Index of a Given Element in a Vector

```
// C++ program to find the index
// of an element in a vector
#include <bits/stdc++.h>

using namespace std;

// print index of element
```

```
    // Condition if element found
    if (it != v.end()) {

        // Calculating the index
        // of element
        int index = it - v.begin();
        cout << index << endl;
    }
    // No such element in vector
    else {
        cout << "-1" << endl;
    }
}

int main()
{
    vector<int> v = { 1, 2, 3, 4, 5, 6 };

    int element = 5;
    print_index(v, element);

    return 0;
}
```

Output

```
4
```

71. Write a Program to Remove Duplicate Elements in an Array Using STL

```
// C++ program to remove the
// duplicate elements from the array
// using STL in C++
#include <bits/stdc++.h>
```

```cpp
// Function to remove duplicate elements
void removeDuplicates(int arr[], int n)
{

    int i;

    set<int> s;

    // Insert the array elements
    // into the set
    for (i = 0; i < n; i++) {
        s.insert(arr[i]);
    }

    set<int>::iterator it;

    // Print the array with duplicates removed
    cout << "\nAfter removing duplicates:\n";
    for (it = s.begin(); it != s.end(); ++it)
        cout << *it << " ";
    cout << '\n';
}

int main()
{
    int arr[] = { 1, 2, 2, 4, 3, 3, 2, 1 };

    int n = sizeof(arr) / sizeof(arr[0]);

    // Print array
    cout << "\nBefore removing duplicates:\n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    removeDuplicates(arr, n);

    return 0;
}
```

```
Before removing duplicates:
1 2 2 4 3 3 2 1
After removing duplicates:
1 2 3 4
```

72. Write a Program to Sort an Array in Descending Order Using STL

```cpp
// C++ program to sort Array
// in descending order
#include <bits/stdc++.h>
using namespace std;

int main()
{
    // Get the array
    int arr[] = { 1, 2, 3, 4, 5, 6 };

    // Compute the sizes
    int n = sizeof(arr) / sizeof(arr[0]);

    // Print the array
    cout << "Array: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    // Sort the array in descending order
    sort(arr, arr + n, greater<int>());

    // Print the array
    cout << "\nDescending Sorted Array:";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

```
Array: 1 2 3 4 5 6
Descending Sorted Array:6 5 4 3 2 1
```

73. Write a Program to Calculate the Frequency of Each Word in the Given String

```cpp
// C++ program to calculate
// frequency of each word
// in given string
#include <bits/stdc++.h>
using namespace std;

// Function to print frequency of each word
void printFrequency(string str)
{
    map<string, int> M;

    string word = "";

    for (int i = 0; i < str.size(); i++) {

        // if element is empty
        if (str[i] == ' ') {

            // If the current word
            // is not found then insert
            // current word with frequency 1
            if (M.find(word) == M.end()) {
                M.insert(make_pair(word, 1));
                word = "";
            }
            else {
                M[word]++;
                word = "";
            }
        }
```

```
    }

    // Storing the last word of the string
    if (M.find(word) == M.end())
        M.insert(make_pair(word, 1));

    // Update the frequency
    else
        M[word]++;

    // Traverse the map
    for (auto& it : M) {
        cout << it.first << " - " << it.second << endl;
    }
}

int main()
{
    string str = "Geeks For Geeks is for Geeks";

    printFrequency(str);
    return 0;
}
```

Output

```
For - 1
Geeks - 3
for - 1
is - 1
```

74. Write a Program to Find k Maximum Elements of an Array in the Original Order

```
// C++ program to find k Maximum elements
#include <bits/stdc++.h>
```

```
void printMax(int arr[], int n, int k)
{
    int result[n], c[n];

    // Coping the array a
    // into c and initialising
    for (int i = 0; i < n; i++) {
        c[i] = arr[i];
        result[i] = 0;
    }

    for (int i = 0; i < k; i++) {

        int maxi = INT_MIN;
        int index;
        for (int j = 0; j < n; j++) {
            if (arr[j] > maxi) {
                maxi = arr[j];
                index = j;
            }
        }
        // Assigning 1 in order
        // to mark the position
        // of all k maximum numbers
        result[index] = 1;
        arr[index] = INT_MIN;
    }

    // Printing elements
    for (int i = 0; i < n; i++) {
        if (result[i] == 1)
            cout << c[i] << " ";
    }
}

int main()
{
    int arr[] = { 50, 8, 45, 12, 25, 40, 84 };
```

```
    return 0;
}
```

Output

```
50 45 84
```

75. Write a Program to Find All Unique Subsets of a Given Set Using STL

```
// C++ code for the above approach:

#include <bits/stdc++.h>
using namespace std;

void solve(vector<int>& arr, int n, set<vector<int> >& ans,
           vector<int> v, int i)
{
    // Base Condition
    if (i >= n) {
        ans.insert(v);
        return;
    }

    solve(arr, n, ans, v, i + 1);

    v.push_back(arr[i]);
    solve(arr, n, ans, v, i + 1);
}

vector<vector<int> > AllSubsets(vector<int> arr, int n)
{

    // Set of vectors to store
    // required unique subsets
    set<vector<int> > ans;
```

```
    solve(arr, n, ans, v, 0);

    // Vector of vectors to store final result
    vector<vector<int> > res;
    while (!ans.empty()) {
        res.push_back(*ans.begin());
        ans.erase(ans.begin());
    }
    return res;
}

// Print Function
void print(int N, vector<int>& A)
{
    vector<vector<int> > result = AllSubsets(A, N);

    // printing the output
    for (int i = 0; i < result.size(); i++) {
        cout << '(';
        for (int j = 0; j < result[i].size(); j++) {
            cout << result[i][j];
            if (j < result[i].size() - 1)
                cout << " ";
        }
        cout << "), ";
    }
    cout << "\n";
}

int main()
{
    int N = 3;
    vector<int> A = { 1, 2, 3 };

    print(N, A);
    return 0;
}
```

```
(), (1), (1 2), (1 2 3), (1 3), (2), (2 3), (3),
```

76. Write a Program to Iterate Over a Queue Without Removing the Element

```cpp
// C++ program to iterate a
// STL Queue by Creating
// copy of given queue
#include <iostream>
#include <queue>
using namespace std;
int main()
{
    queue<int> q;

    // Inserting elements in queue
    q.push(1);
    q.push(2);
    q.push(3);
    q.push(4);
    q.push(5);

    // Copy queue
    queue<int> copy_queue = q;

    cout << "Queue elements :\n";

    while (!copy_queue.empty()) {
        cout << copy_queue.front() << " ";

        copy_queue.pop();
    }

    return 0;
}
```

```
Queue elements :
1 2 3 4 5
```

77. Write a Program for the Implementation of Stacks Using an Array

```cpp
// C++ Program to implement stacks using array
#include <iostream>
using namespace std;

// Maximum size of stack

#define MAX 100

class Stack {
private:
    int top;
    // Array to store stack elements
    int arr[MAX];

public:
    // Constructor to initialize top as -1
    Stack() { top = -1; }

    // Function to push an element to the stack
    void push(int x)
    {
        if (top == MAX - 1) {
            cout << "Stack overflow" << endl;
            return;
        }
        arr[++top] = x;
    }

    // Function to pop an element from the stack
    int pop()
```

```
            return -1;
        }
        return arr[top--];
    }

    // Function to check if stack is empty
    bool isEmpty() { return (top == -1); }

    // Function to return the top element of the stack
    int peek()
    {
        if (top == -1) {
            cout << "Stack is empty" << endl;
            return -1;
        }
        return arr[top];
    }
};

int main()
{
    Stack s;
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);
    s.push(5);

    cout << "Popped element: " << s.pop() << endl;
    cout << "Top element: " << s.peek() << endl;

    return 0;
}
```

Output

```
Popped element: 5
```

78. Write a Program for the Implementation of a Queue Using an Array

```cpp
// C++ Program to implement queue using array
#include <iostream>
using namespace std;

// Maximum size of the queue
#define MAX 100

class Queue {
private:
    int front, rear;
    // Array to store queue elements
    int arr[MAX];

public:
    // Constructor to initialize front and rear as -1

    Queue() { front = rear = -1; }
    // Function to add an element to the queue
    void enqueue(int x)
    {
        if (rear == MAX - 1) {
            cout << "Error: Queue overflow" << endl;
            return;
        }
        arr[++rear] = x;
        if (front == -1)
            front = 0;
    }

    // Function to remove an element from the queue
    int dequeue()
    {
        if (front == -1) {
            cout << "Queue underflow" << endl;
```

```
        if (front == rear)
            front = rear = -1;
        else
            front++;
        return x;
    }

    // Function to check if queue is empty
    bool isEmpty() { return (front == -1); }

    // Function to return the front element of the queue
    int peek()
    {
        if (front == -1) {
            cout << "Queue is empty" << endl;
            return -1;
        }
        return arr[front];
    }
};

int main()
{
    Queue q;
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    q.enqueue(4);
    q.enqueue(5);

    cout << "Dequeued element: " << q.dequeue() << endl;
    cout << "Front element: " << q.peek() << endl;

    return 0;
}
```

Output

79. Write a Program Implementation of Stacks Using a Queue

```cpp
// C++ Program to implement
// Stack using queue
#include <bits/stdc++.h>

using namespace std;

class Stack {
    queue<int> q1, q2;

public:
    void push(int x)
    {
        // Push x first in empty q2
        q2.push(x);

        // Push all the remaining
        // elements in q1 to q2.
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }

        // swap the names of two queues
        queue<int> q = q1;
        q1 = q2;
        q2 = q;
    }

    void pop()
    {
        // if no elements are there in q1
        if (q1.empty())
            return;
        q1.pop();
    }
```

```
        if (q1.empty())
            return -1;
        return q1.front();
    }

    int size() { return q1.size(); }
};

int main()
{
    Stack s;

    // Inserting elements in Stack
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);

    cout << "Size: " << s.size() << endl;
    cout << s.top() << endl;
    s.pop();
    cout << s.top() << endl;
    s.pop();
    cout << s.top() << endl;

    cout << "Size: " << s.size() << endl;
    return 0;
}
```

Output

```
Size: 4
4
3
2
Size: 2
```

```cpp
// C++ Program to Implement
// stack using list
#include <bits/stdc++.h>

using namespace std;

// Template declared
template <typename T> class Stack {
public:
    list<T> l;
    int cs = 0;
    // current size of the stack

    // pushing an element into the stack
    void push(T d)
    {
        cs++;
        // increasing the current size of the stack
        l.push_front(d);
    }

    // popping an element from the stack
    void pop()
    {
        if (cs <= 0) {
            // cannot pop us stack does not contain an
            // elements
            cout << "Stack empty" << endl;
        }
        else {
            // decreasing the current size of the stack
            cs--;
            l.pop_front();
        }
    }

    // if current size is 0 then stack is empty
    bool empty() { return cs == 0; }
```

```
    {
        // getting the size of the stack
        return cs;
    }

    // printing the elements of the stack
    void print()
    {
        for (auto x : l) {
            cout << x << endl;
        }
    }
};
int main()
{
    // Inserting elements in stack
    Stack<int> s;
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);

    cout << "Size: " << s.size() << endl;
    cout << "Top element:" << s.top() << endl;
    s.pop();
    cout << "Top element:" << s.top() << endl;
    s.pop();
    cout << "Top element:" << s.top() << endl;
    cout << "Size:" << s.size() << endl;
    return 0;
}
```

Output

```
Size: 4
Top element:4
Top element:3
```

81. Write a Program to Determine Array is a Subset of Another Array or Not

```cpp
// C++ Program to check if
// Array is a subset of another array or not

#include <iostream>
using namespace std;

bool isSubset(int arr1[], int arr2[], int m, int n)
{
    int i = 0;
    int j = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            if (arr2[i] == arr1[j])
                break;
        }

        if (j == m)
            return 0;
    }

    return 1;
}

int main()
{
    int arr1[] = { 1, 11, 31, 21, 30, 17 };
    int arr2[] = { 11, 30, 17, 1 };

    int m = sizeof(arr1) / sizeof(arr1[0]);
    int n = sizeof(arr2) / sizeof(arr2[0]);

    if (isSubset(arr1, arr2, m, n))
        cout << "arr2 is subset of arr1 ";
    else
```

```
}
```

Output

```
arr2 is subset of arr1
```

82. Write a Program for Finding the Circular Rotation of an Array by K Positions

```
// C++ Program for Finding
// Circular rotation of an array
// by K positions
#include <iostream>
using namespace std;

void Rotate(int arr[], int k, int n)
{
    // temp array
    int temp[n];

    // Keepig track of the current index
    // of temp[]
    int t = 0;

    // Storing the n - d elements of
    // array arr[] to the front of temp[]
    for (int i = k; i < n; i++) {
        temp[t] = arr[i];
        t++;
    }

    // Storing the first d elements of array arr[]
    // into temp
    for (int i = 0; i < k; i++) {
        temp[t] = arr[i];
        t++;
```

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

void print_array(int arr[], int n)
{
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };
    int N = sizeof(arr) / sizeof(arr[0]);
    int k = 2;

    // Function calling
    Rotate(arr, k, N);
    print_array(arr, N);

    return 0;
}
```

Output

```
3 4 5 1 2
```

83. Write a Program to Sort the First Half in Ascending Order and the Second Half in Descending

```
// C++ Program to Sort first half
// in ascending order and second half in descending
#include <iostream>
using namespace std;
```

```
    int temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n / 2; j++) {
            if (a[j] > a[j + 1]) {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }

        for (int j = n / 2; j < n - 1; j++) {
            if (a[j] < a[j + 1]) {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }

    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    int len = sizeof(arr) / sizeof(arr[0]);

    ascDecFunc(arr, len);

    return 0;
}
```

Output

```
1 2 3 4 8 7 6 5
```

```
// of a string using Last to First
#include <iostream>
using namespace std;

void reverse(string str)
{
    for (int i = str.length() - 1; i >= 0; i--)
        cout << str[i];
}

int main(void)
{
    string str = "GeeksforGeeks";

    reverse(str);

    return (0);
}
```

Output

```
skeeGrofskeeG
```

85. Write a Program to Print All Permutations of a String Using Recursion

```
// C++ Program to Print
// Permutaions of string
#include <iostream>
#include <string>
using namespace std;

void permute(string s, string answer)
{
    if (s.length() == 0) {
        cout << answer << endl;
```

```
        char ch = s[i];
        string left = s.substr(0, i);
        string right = s.substr(i + 1);
        string result = left + right;
        permute(result, answer + ch);
    }
}

int main()
{
    string s = "ABC";
    string answer = "";

    permute(s, answer);
    return 0;
}
```

Output

```
ABC
ACB
BAC
BCA
CAB
CBA
```

86. Write a Program to Print All Permutations of a Given String in Lexicographically Sorted Order

```
// C++ Program to Print all permutations
// Of a given string in lexicographically sorted order
#include <bits/stdc++.h>
using namespace std;

int compare(const void* a, const void* b)
{
```

```
void swap(char* a, char* b)
{
    char t = *a;
    *a = *b;
    *b = t;
}

int findCeil(char str[], char first, int l, int h)
{
    int ceilIndex = l;

    for (int i = l + 1; i <= h; i++)
        if (str[i] > first && str[i] < str[ceilIndex])
            ceilIndex = i;

    return ceilIndex;
}

void Permutations(char str[])
{
    int size = strlen(str);

    qsort(str, size, sizeof(str[0]), compare);

    bool isFinished = false;
    while (!isFinished) {
        cout << str << endl;
        int i;

        for (i = size - 2; i >= 0; --i)
            if (str[i] < str[i + 1])
                break;

        if (i == -1)
            isFinished = true;
        else {
            int ceilIndex
                = findCeil(str, str[i], i + 1, size - 1);
```

```
        }
    }
}

int main()
{
    char str[] = "XYZ";
    Permutations(str);
    return 0;
}
```

Output

```
XYZ
XZY
YXZ
YZX
ZXY
ZYX
```

87. Write a Program to Remove Brackets From an Algebraic Expression

```
// C++ Program to remove brackets from an algebraic
#include <iostream>
#include <string>

using namespace std;

string remove_brackets(string str)
{
    string result = "";
    for (char c : str) {
        if (c != '(' && c != ')') {
            result += c;
```

```
int main()
{
    string str = "Geeks)(for)(geeks";

    cout << "Without brackets: " << remove_brackets(str)
         << endl;

    return 0;
}
```

88. Program to Perform Insert, Delete, and Print Operations Singly Linked List

```
// C++ Program to perform insertion, deletion, and p
// operations in LL
#include <iostream>

using namespace std;

struct Node {
    int data;
    Node* next;
};

Node* head = nullptr;

// Function to insert node in LL
void insert(int data)
{
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}
// function to delete node of LL
```

```
        head = temp->next;
        delete temp;
        return;
    }
    while (temp != nullptr && temp->data != data) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == nullptr)
        return;
    prev->next = temp->next;
    delete temp;
}

// function to print LL
void printList()
{
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main()
{
    insert(1);
    insert(2);
    insert(3);
    insert(4);
    insert(5);
    cout << "Linked List is \n";

    printList();
    deleteNode(3);
    cout << "Linked List after deletion of 3: ";
    printList();
```

Output

```
Linked List is
5 4 3 2 1
Linked List after deletion of 3: 5 4 2 1
```

89. Program to Perform Insert, Delete, and Print Operations Doubly Linked List

```cpp
// C++ Program to implement insert, delete, and prin
// operation in doubly linked list
#include <iostream>

using namespace std;

struct Node {
    int data;
    Node* next;
    Node* prev;
};

class DoublyLinkedList {
private:
    Node* head;

public:
    DoublyLinkedList()
        : head(NULL)
    {
    }

    void insertAtStart(int data)
    {
        Node* newNode = new Node;
```

```
        if (head != NULL)
            head->prev = newNode;

        head = newNode;
    }

    void deleteNode(int data)
    {
        Node* temp = head;
        while (temp != NULL && temp->data != data)
            temp = temp->next;

        if (temp == NULL)
            return;

        if (temp->prev != NULL)
            temp->prev->next = temp->next;
        else
            head = temp->next;

        if (temp->next != NULL)
            temp->next->prev = temp->prev;

        delete temp;
    }

    void printList()
    {
        Node* temp = head;
        while (temp != NULL) {
            std::cout << temp->data << " ";
            temp = temp->next;
        }
        std::cout << std::endl;
    }
};

int main()
```

```
    dll.insertAtStart(1);
    dll.insertAtStart(2);
    dll.insertAtStart(3);
    dll.insertAtStart(4);
    dll.insertAtStart(5);

    std::cout << "Original Doubly Linked List: ";
    dll.printList();

    dll.deleteNode(2);

    std::cout << "Doubly Linked List after deletion: ";
    dll.printList();

    return 0;
}
```

Output

```
Original Doubly Linked List: 5 4 3 2 1
Doubly Linked List after deletion: 5 4 3 1
```

90. Program to Perform Insert, Delete, and Print Operations Circular Linked List

```
// C++ Program to implement insert, delete, and prin
// circular linked list
#include <iostream>

struct Node {
    int data;
    Node* next;
};

class CircularLinkedList {
private:
```

```
    CircularLinkedList()
        : head(NULL)
    {
    }

    void insertAtStart(int data)
    {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->next = head;

        if (head == NULL) {
            head = newNode;
            newNode->next = head;
        }
        else {
            Node* temp = head;
            while (temp->next != head)
                temp = temp->next;
            temp->next = newNode;
            head = newNode;
        }
    }

    void deleteNode(int data)
    {
        Node* temp = head;
        if (temp == NULL)
            return;

        if (temp->next == head) {
            head = NULL;
            delete temp;
            return;
        }

        Node* prev = NULL;
        while (temp->next != head && temp->data != data) {
```

```
        if (temp->data != data)
            return;

        prev->next = temp->next;
        if (temp == head)
            head = temp->next;
        delete temp;
    }

    void printList()
    {
        Node* temp = head;
        while (temp->next != head) {
            std::cout << temp->data << " ";
            temp = temp->next;
        }
        std::cout << temp->data << std::endl;
    }
};

int main()
{
    CircularLinkedList cll;

    cll.insertAtStart(1);
    cll.insertAtStart(2);
    cll.insertAtStart(3);
    cll.insertAtStart(4);
    cll.insertAtStart(5);

    std::cout << "Original Circular Linked list ";
    cll.printList();

    cll.deleteNode(2);

    std::cout << "Circular Linked List after deletion ";
    cll.printList();
```

Output

```
Original Circular Linked list 5 4 3 2 1
Circular Linked List after deletion 5 4 3 1
```

91. Program for Inorder Traversal in a Binary Tree

```cpp
// C++ program Inorder Traversal
#include <bits/stdc++.h>
using namespace std;

/* A binary tree node has data, pointer to left child
and a pointer to right child */

struct Node {
    int data;
    struct Node *left, *right;
};

// Utility function to create a new tree node
Node* newNode(int data)
{
    Node* temp = new Node;
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct Node* node)
{
    if (node == NULL)
        return;

    /* first recur on left child */
```

```
    cout << node->data << " ";

    /* now recur on right child */
    printInorder(node->right);
}

int main()
{
    struct Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    // Function call
    cout << "Inorder traversal of binary tree is \n";
    printInorder(root);

    return 0;
}
```

Output

```
Inorder traversal of binary tree is
4 2 5 1 3
```

92. Program to Find All its Subsets From a Set of Positive Integers

```
// C++ Program to find all subsets from the given se ✕ of ▷
// positive integers
#include <bits/stdc++.h>

using namespace std;

void find_subset(vector<int>& A, vector<vector<int> >& ans,
                 vector<int>& subset, int index)
```

```
        subset.push_back(A[i]);
        find_subset(A, ans, subset, i + 1);
        subset.pop_back();
    }

    return;
}

vector<vector<int> > subsets(vector<int>& A)
{
    vector<int> subset;
    vector<vector<int> > ans;

    int index = 0;
    find_subset(A, ans, subset, index);

    return ans;
}

int main()
{
    vector<int> array = { 1, 2, 3, 4, 5 };

    vector<vector<int> > ans = subsets(array);

    for (int i = 0; i < ans.size(); i++) {
        for (int j = 0; j < ans[i].size(); j++)
            cout << ans[i][j] << " ";
        cout << endl;
    }

    return 0;
}
```

Output

```
1 2 3 4 5
1 2 3 5
1 2 4
1 2 4 5
1 2 5
1 3
1 3 4
1 3 4 5
1 3 5
1 4
1 4 5
1 5
2
2 3
2 3 4
2 3 4 5
2 3 5
2 4
2 4 5
2 5
3
3 4
3 4 5
3 5
4
4 5
5
```

93. Program for Preorder Traversal in a Binary Tree

```cpp
// C++ program for preorder traversal
#include <bits/stdc++.h>
using namespace std;

/* A binary tree node has data, pointer to left child
```

```
    struct Node *left, *right;
};

// Utility function to create a new tree node
Node* newNode(int data)
{
    Node* temp = new Node;
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

/* Given a binary tree, print its nodes in preorder*/
void printPreorder(struct Node* node)
{
    if (node == NULL)
        return;

    /* first print data of node */
    cout << node->data << " ";

    /* then recur on left subtree */
    printPreorder(node->left);

    /* now recur on right subtree */
    printPreorder(node->right);
}

int main()
{
    struct Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    // Function call
    cout << "Preorder traversal of binary tree is \n";
```

```
}
```

Output

```
Preorder traversal of binary tree is
1 2 4 5 3
```

94. Program for Postorder Traversal in a Binary Tree

```cpp
// C++ program for post order traversal
#include <bits/stdc++.h>
using namespace std;

/* A binary tree node has data, pointer to left child
and a pointer to right child */
struct Node {
    int data;
    struct Node *left, *right;
};

// Utility function to create a new tree node
Node* newNode(int data)
{
    Node* temp = new Node;
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

/* Given a binary tree, print its nodes according to the
"bottom-up" postorder traversal. */
void printPostorder(struct Node* node)
{
    if (node == NULL)
        return;
```

```
    // then recur on right subtree
    printPostorder(node->right);

    // now deal with the node
    cout << node->data << " ";
}

int main()
{
    struct Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    // Function call
    cout << "Postorder traversal of binary tree is \n";
    printPostorder(root);

    return 0;
}
```

Output

```
Postorder traversal of binary tree is
4 5 2 3 1
```

95. Program for Level-Order Traversal in a Binary Tree

```
// C++ program for post order traversal
#include <bits/stdc++.h>
using namespace std;

/* A binary tree node has data, pointer to left child
and a pointer to right child */
struct Node {
```

```
// Utility function to create a new tree node
Node* newNode(int data)
{
    Node* temp = new Node;
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

/* Given a binary tree, print its nodes according to the
"bottom-up" postorder traversal. */
void printLevelOrder(struct Node* node)
{
    if (node == NULL)
        return;
    queue<struct Node*> q;
    while (node) {
        if (node->left) {
            q.push(node->left);
        }
        if (node->right) {
            q.push(node->right);
        }
        cout << node->data << " ";
        node = q.front();
        q.pop();
    }
}

int main()
{
    struct Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
```

```
    return 0;
}
```

Output

```
Levelorder traversal of binary tree is
1 2 3 4 5
```

96. Write a Program for the Top View of a Binary Tree

```
// C++ program for top view of a binary tree
#include <bits/stdc++.h>
using namespace std;

// node data type
struct Node {
    Node* left;
    Node* right;
    int hd;
    int data;
};

// utility function to create new node
Node* newNode(int key)
{
    Node* node = new Node();
    node->left = node->right = NULL;
    node->data = key;
    return node;
}

// function to print top view of a binary tree
void topView(Node* root)
{
    if (root == NULL) {
```

```
    map<int, int> m;
    int hd = 0;
    root->hd = hd;

    q.push(root);

    while (q.size()) {
        hd = root->hd;

        if (m.count(hd) == 0)
            m[hd] = root->data;
        if (root->left) {
            root->left->hd = hd - 1;
            q.push(root->left);
        }
        if (root->right) {
            root->right->hd = hd + 1;
            q.push(root->right);
        }
        q.pop();
        root = q.front();
    }

    // printing top view
    for (auto i = m.begin(); i != m.end(); i++) {
        cout << i->second << " ";
    }
}

// Driver code
int main()
{
    // new binary tree
    Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->right->right = newNode(5);
```

```
}
```

Output

```
4 2 1 3 5
```

97. Write a Program to Print the Bottom View of a Binary Tree

```cpp
// C++ program for bottom view of a binary tree
#include <bits/stdc++.h>
using namespace std;

// node data type
struct Node {
    Node* left;
    Node* right;
    int data;
    int hd;
};

// utility function for new node
Node* newNode(int key)
{
    Node* node = new Node();
    node->data = key;
    node->left = node->right = NULL;
    node->hd = 0;

    return node;
}

// function to print bottom view
void bottomView(Node* root)
{
    if (root == NULL)
        return;
```

```
    int hd = 0;

    root->hd = hd;
    q.push(root);

    while (!q.empty()) {
        Node* temp = q.front();
        q.pop();

        hd = temp->hd;

        m[hd] = temp->data;

        if (temp->left != NULL) {
            temp->left->hd = hd - 1;
            q.push(temp->left);
        }

        if (temp->right != NULL) {
            temp->right->hd = hd + 1;
            q.push(temp->right);
        }
    }

    // printing top view
    for (auto i = m.begin(); i != m.end(); ++i)
        cout << i->second << " ";
}

// Driver Code
int main()
{
    Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->right->right = newNode(5);
```

```
}
```

Output

```
4 2 1 3 5
```

98. Write a Program to Print the Left View of a Binary Tree

```cpp
// C++ program to print left view of a binary tree
#include <bits/stdc++.h>
using namespace std;

// node datatype
struct Node {
    Node* left;
    Node* right;
    int data;
};

// utility function to create a new node
Node* newNode(int key)
{
    Node* node = new Node;
    node->data = key;
    node->left = node->right = NULL;
    return node;
}

// function to print left view
void leftView(Node* root)
{
    if (!root) {
        return;
    }

    queue<Node*> q;
```

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

        for (int i = 1; i <= n; i++) {
            Node* temp = q.front();
            q.pop();

            if (i == 1)
                cout << temp->data << " ";

            if (temp->left != NULL)
                q.push(temp->left);

            if (temp->right != NULL)
                q.push(temp->right);
        }
    }
}

// Driver code
int main()
{

    Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->right->right = newNode(5);

    leftView(root);
}
```

Output

```
1 2 4
```

99. Write a Program to Print the Right View of the Binary Tree

```
using namespace std;

// node data type
struct Node {
    Node* left;
    Node* right;
    int data;
};

// utility function to create new node
Node* newNode(int key)
{
    Node* node = new Node;
    node->data = key;
    node->left = node->right = NULL;
    return node;
}

// function to print right view of a binary tree
void rightView(Node* root)
{
    if (root == NULL) {
        return;
    }

    queue<Node*> q;
    q.push(root);

    while (!q.empty()) {

        int n = q.size();

        while (n--) {

            Node* x = q.front();
            q.pop();

            if (n == 0) {
```

```
                q.push(x->left);
            }
            if (x->right) {
                q.push(x->right);
            }
        }
    }
}

// Driver code
int main()
{

    Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->right->right = newNode(5);

    rightView(root);
}
```

Output

```
1 3 5
```

100. Write a Program for the Conversion of Infix Expression to Postfix Expression

```
// C++ program for infix expression to postfix expre
// conversion
#include <bits/stdc++.h>
using namespace std;

// infix to postfix conversion
// supported operators => + - * /
```

```
    string postfix;
    // operator stack
    stack<char> ope_stack;
    // operator precedence map
    unordered_map<char, int> pre = { { '+', 1 },
                                     { '-', 1 },
                                     { '*', 2 },
                                     { '/', 2 },
                                     { '(', 0 } };

    int length = expression.length();

    for (int i = 0; i < length; i++) {
        char c = expression[i];
        // checking operands
        if ((c >= 92 && c <= 122) || (c >= 48 && c <= 57)) {
            postfix.push_back(c);
        }
        // checking braces
        else if (c == '(') {
            ope_stack.push(c);
        }
        else if (c == ')') {
            while (ope_stack.top() != '(') {
                postfix.push_back(ope_stack.top());
                ope_stack.pop();
            }
            ope_stack.pop();
        }
        // checking operators
        else {
            while (!ope_stack.empty()
                   && pre.at(c) < pre.at(ope_stack.top())) {
                postfix.push_back(ope_stack.top());
                ope_stack.pop();
            }
            ope_stack.push(c);
        }
```

```
    while (!ope_stack.empty()) {
        postfix.push_back(ope_stack.top());
        ope_stack.pop();
    }

    return postfix;
}

// driver code
int main()
{
    string s = "a*b+(c-d)";
    cout << infixToPostfix(s);

    return 0;
}
```

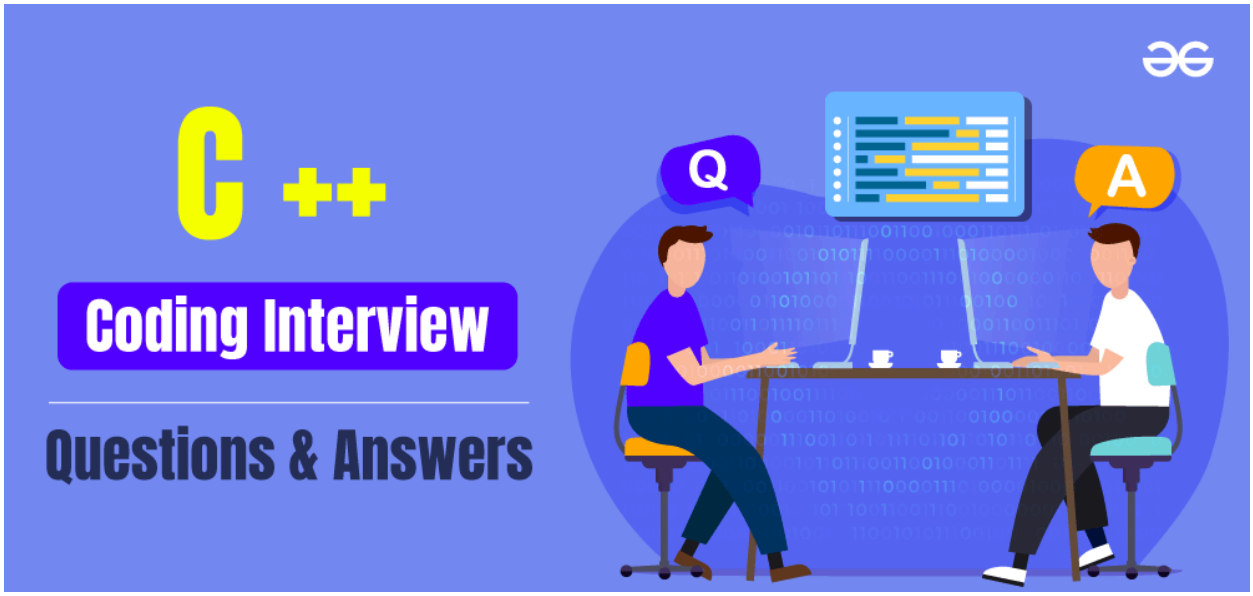
Output

```
ab*cd-+
```

Conclusion

Coding interviews can be challenging task, but they are also a great opportunity to test your skills and knowledge. By practicing with commonly ask **C++ coding interview questions**, you can increase your chances of success.

Remember to stay calm, communicate your thought process clearly, and don't be afraid to ask for clarification if needed. With hard work and preparation, you can ace your interview and land your dream job!



Comment More info

Campus Training Program

Next Article

Python Interview Questions and Answers

Similar Reads

1. AMD Interview Experience for Internship (On-Campus)
2. 7 Essential C++ Concepts for Every Developer

6. JP Morgan Interview Questions and Answers for Technical Profiles
7. Top Interview Problems Asked in 2024 (Topic Wise)
8. Top Interview Problems Asked in 2024 (Company Wise)
9. Top 10 Reasons to Learn C++ in 2025
10. SAP Labs Interview Questions | Set 8 (Bangalore)

PHP
GoLang
SQL
R Language
Android

Basic DSA Problems
DSA Roadmap
DSA Interview Questions
Competitive Programming

Data Science & ML

Data Science With Python
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
NodeJs
Bootstrap
Tailwind CSS

Python Tutorial

Python Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB