**C++ Programming Code:**

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

**Code:**

**#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);**

**}**

Q. **Write a Program Implementation of Stacks Using a Queue**

Code:

#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();

}

int top()

{

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;

}

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

Code:

// C++ program to find k Maximum elements

#include <bits/stdc++.h>

using namespace std;

// Function to print k Maximum elements

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 };

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

int k = 3;

printMax(arr, n, k);

return 0;

}

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

Code:

// 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 = "";

}

}

else

word += str[i];

}

// 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;

}

Q. **Write a Program to Implement Binary Search**

Code:

// 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)

return binarySearch(arr, l, mid - 1, x);

return binarySearch(arr, mid + 1, r, 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;

}

Q . **Write a Program to Implement Linear Search**

Code:

// 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 main()

{

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;

}

Q. **Write a Program to Implement Quick Sort**

Code:

// C++ Program to implement

// QuickSort

#include <iostream>

using namespace std;

// Swap elements

void swap(int\* a, int\* b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

// 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;

for (i = 0; i < size; i++)

cout << arr[i] << " ";

cout << endl;

}

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;

}

Q. **Write a Program to Implement Merge Sort**

Code:

// 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]

<= rightArray[indexOfSubArrayTwo]) {

array[indexOfMergedArray]

= 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);

}

// Print Array

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;

}

Q. **Write a Program to Implement Selection Sort**

Code:

// C++ program to implement

// Selection sort

#include <iostream>

using namespace std;

// Swap function

void swap(int\* p, int\* q)

{

int temp = \*p;

\*p = \*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);

return 0;

}

Q. **Write a Program to Implement Insertion Sort**

**Code:**

**// 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];**

**j = j - 1;**

**}**

**arr[j + 1] = key;**

**}**

**}**

**// 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;**

**}**

**Q. Write a Program to Implement Bubble Sort**

**Code:**

**// 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;**

**for (i = 0; i < size; i++)**

**cout << arr[i] << " ";**

**cout << endl;**

**}**

**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;**

**}**

**Q. Write a Program to Implement Single-Level Inheritance**

**Code:**

**// 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;**

**public:**

**void set\_s(int id, char\* name, char\* course, 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;**

**}**

**Q. Write a Program to Implement the Concept of Abstraction**

**Code:**

**// 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;**

**}**

**void display()**

**{**

**cout << "p = " << p << endl;**

**cout << "q = " << q << endl;**

**}**

**};**

**int main()**

**{**

**implementAbstraction obj;**

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

**obj.display();**

**return 0;**

**}**

**Q. Write a Program to Implement the Use of Encapsulation**

**Code:**

**// 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; }**

**// function to return value of**

**// variable x**

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

**};**

**int main()**

**{**

**Encapsulation obj;**

**obj.setter(13);**

**cout << obj.getter();**

**return 0;**

**}**

**Q. Write a Program for Decimal to Binary Conversion**

**Code:**

**// 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;**

**}**

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

**Code:**

**// C++ program to find**

**// maximum and minimum of**

**// Two numbers without using**

**// loop and conditions**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**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;**

**}**

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

**Code:**

**// C++ Program to count the sume of numbers in a string**

**#include <iostream>**

**#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;**

**}**

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

**Code:**

**// 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;**

**}**

**Q. Write a Program to Print the Pascal Triangle**

**Code:**

**// 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**

**arr[line][i] = arr[line - 1][i - 1]**

**+ arr[line - 1][i];**

**cout << arr[line][i] << " ";**

**}**

**cout << "\n";**

**}**

**}**

**int main()**

**{**

**int n = 6;**

**printPascal(n);**

**return 0;**

**}**

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

Code:

// 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);

return 0;

}