## 1. Vector Basic Operations

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

int main() {
    vector<int> v = {10, 20, 30};
    cout << "Size: " << v.size() << endl;
    v.push_back(40);
    v.insert(v.begin() + 1, 15);
    v[2] = 25;
    v.erase(v.begin() + 3);
    v.pop_back();
    cout << "Final vector: ";
    for (int x : v) cout << x << " ";
}
```

Demonstrates vector size, insert, update, delete, and display operations in C++ using STL vector.

## 2. 2D Vector Insertion & Display

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<vector<int>> v2d = {{1,2,3},{4,5},{6}};
    v2d[1].push_back(9);
    v2d.push_back({7,8,9});
    for (auto row : v2d) {
        for (auto col : row) cout << col << " ";
        cout << endl;
    }
}
```

Shows how to insert elements and display contents in a 2D vector structure.

## 3. Sorting a Vector

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> v = {5,1,4,2,3};
    sort(v.begin(), v.end());
    cout << "Ascending: ";
    for (int x : v) cout << x << " ";
    cout << endl;
    sort(v.begin(), v.end(), greater<int>());
    cout << "Descending: ";
    for (int x : v) cout << x << " ";
}
```

Sorting a vector in ascending and descending order using the STL sort() function.

## 4. Trailing Zeroes in Factorial

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n; cin >> n;
    int count = 0;
    for (int i = 5; n / i >= 1; i *= 5) count += n / i;
    cout << count;
}
```

Counts trailing zeroes in factorial using powers of 5.

## 5. Count Digits in Factorial

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

```
int main() {
    int n; cin >> n;
    if (n == 0 || n == 1) { cout << 1; return 0; }
    double digits = 0;
    for (int i = 2; i <= n; i++) digits += log10(i);
    cout << floor(digits) + 1;
}
```

Calculates the number of digits in factorial using logarithmic approach.

## 6. Only Pluses Problem

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> v = {1,3,5};
    int mx = *max_element(v.begin(), v.end());
    int moves = 0;
    for (int x : v) moves += mx - x;
    cout << moves;
}
```

Finds minimum '+' operations needed to make all elements equal.

## 7. Linear Search

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> a = {5,3,8,2};
    int key = 8;
    for (int i = 0; i < a.size(); i++)
        if (a[i] == key) { cout << i; return 0; }
    cout << "Not found";
}
```

Simple linear search implementation.

## 8. Binary Search

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> a = {2,3,4,10,40};
    int key = 10;
    int l = 0, r = a.size() - 1;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (a[mid] == key) { cout << mid; return 0; }
        else if (a[mid] < key) l = mid + 1;
        else r = mid - 1;
    }
    cout << "Not found";
}
```

Binary search algorithm for sorted arrays.

## 9. Bubble Sort

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> a = {5,1,4,2,8};
    for (int i = 0; i < a.size() - 1; i++)
        for (int j = 0; j < a.size() - i - 1; j++)
            if (a[j] > a[j + 1]) swap(a[j], a[j + 1]);
    for (int x : a) cout << x << " ";
}
```

Bubble sort algorithm - repeatedly swaps adjacent elements.

## 10. Selection Sort

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> a = {29,10,14,37,13};
    for (int i = 0; i < a.size() - 1; i++) {
        int minIdx = i;
        for (int j = i + 1; j < a.size(); j++)
            if (a[j] < a[minIdx]) minIdx = j;
        swap(a[i], a[minIdx]);
    }
    for (int x : a) cout << x << " ";
}
```

Selection sort - repeatedly selects the smallest element.

## 11. Insertion Sort

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> a = {12,11,13,5,6};
    for (int i = 1; i < a.size(); i++) {
        int key = a[i], j = i - 1;
        while (j >= 0 && a[j] > key) a[j + 1] = a[j--];
        a[j + 1] = key;
    }
    for (int x : a) cout << x << " ";
}
```

Insertion sort algorithm for small datasets.

## 12. Kadane’s Algorithm (Maximum Subarray Sum)

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> arr = {-2,1,-3,4,-1,2,1,-5,4};
    int maxSum = INT_MIN, curr = 0;
    for (int x : arr) {
        curr = max(x, curr + x);
        maxSum = max(maxSum, curr);
    }
    cout << maxSum;
}
```

Kadane’s algorithm efficiently finds the maximum sum of any contiguous subarray.