**LAB 1**

**Bubble Sort**

**1. #include <iostream>**

**2. using namespace std;**

**3.**

**4. void printCount(int count) {**

**5.     cout << "Number of iterations: " << count << endl;**

**6. }**

**7.**

**8. void bubbleSort(int arr[], int n) {**

**9.     int iterationCount = 0;**

**10.     bool swapped;**

**11.**

**12.     for (int i = 0; i < n - 1; i++) {**

**13.         swapped = false;**

**14.         for (int j = 0; j < n - i - 1; j++) {**

**15.             iterationCount++;**

**16.             if (arr[j] > arr[j + 1]) {**

**17.                 int temp = arr[j];**

**18.                 arr[j] = arr[j + 1];**

**19.                 arr[j + 1] = temp;**

**20.                 swapped = true;**

**21.             }**

**22.         }**

**23.         if (!swapped) {**

**24.             cout << "Best case" << endl;**

**25.             break;**

**26.         }**

**27.     }**

**28.     printCount(iterationCount);**

**29. }**

**30.**

**31. void printArray(int arr[], int size) {**

**32.     for (int i = 0; i < size; i++)**

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

**34.     cout << endl;**

**35. }**

**36.**

**37. int main() {**

**38.     int n;**

**39.     cout << "Enter the size of the array: ";**

**40.     cin >> n;**

**41.**

**42.     int arr[n];**

**43.     cout << "Enter " << n << " elements of the array: ";**

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

**45.         cin >> arr[i];**

**46.     }**

**47.**

**48.     cout << "Original Array: ";**

**49.     printArray(arr, n);**

**50.**

**51.     bubbleSort(arr, n);**

**52.**

**53.     cout << "Sorted Array: ";**

**54.     printArray(arr, n);**

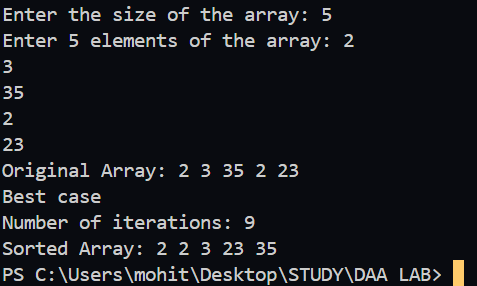
**55.**

**56.     return 0;**

**57. }**

**58.**

|  |  |  |
| --- | --- | --- |
|  | **LOOPS** | **COMP** |
| **BEST** | **1** | **4** |
| **WORSE** | **4** | **10** |
| **AVG** | **3** | **9** |

****

**Insertion Sort**

**1. #include <iostream>**

**2. #include <vector>**

**3.**

**4. using namespace std;**

**5.**

**6. void insertionSort(vector<int> & arr , int& iterations) {**

**7.   int n = arr.size();**

**8.   for (int i = 1; i < n; ++i) {**

**9.     int key = arr[i];**

**10.     int j = i - 1;**

**11.     while (j >= 0 && arr[j] > key) {**

**12.       arr[j + 1] = arr[j];**

**13.       --j;**

**14.       ++iterations;**

**15.     }**

**16.     arr[j + 1] = key;**

**17.   }**

**18. }**

**19.**

**20. int main(){**

**21.   int n;**

**22.   cout << "Enter the number of elements: ";**

**23.   cin >> n;**

**24.**

**25.   vector<int> arr(n);**

**26.   cout << "Enter the elements: ";**

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

**28.     cin >> arr[i];**

**29.   }**

**30.**

**31.   int iterations = 0;**

**32.   insertionSort(arr, iterations);**

**33.**

**34.   cout << "Sorted array: ";**

**35.   for (int i = 0; i < arr.size(); ++i) {**

**36.     cout << arr[i] << " ";**

**37.   }**

**38.   cout << endl;**

**39.**

**40.   cout << "Number of iterations: " << iterations << endl;**

**41.   return 0;**

**42. }**

**43.**

|  |  |  |
| --- | --- | --- |
|  | **LOOPS** | **COMP** |
| **BEST** | **10** | **10** |
| **WORSE** | **10** | **10** |
| **AVG** | **10** | **10** |

**A computer screen shot of a black background

Description automatically generated**

**LEETCODE PROBLEMS1.**

**Given an array Of integers nums and an integer target, return indices Of the two numbers such that they add up to target.You may assume that each input would have exactly one solution and you may not use the same element twice. You can retum the answer in any order.**

**1. class Solution {**

**2. public:**

**3.     vector<int> twoSum(vector<int>& nums, int target) {**

**4.         int n = nums.size();**

**5.         for(int i=0;i<n;i++){**

**6.             for(int j=i+1;j<n;j++){**

**7.                 if(nums[i]+nums[j]==target){**

**8.                     return{i,j};**

**9.                 }**

**10.             }**

**11.         }**

**12.         return {-1,-1};**

**13.**

**14.     }**

**15. };**

**16.**

**A screenshot of a computer

Description automatically generated**

**1class Solution {**

**2. public:**

**3. vector<vector<int>> threeSum(vector<int> &nums) {**

**4. sort(begin(nums), end(nums));**

**5. vector<vector<int>> result;**

**6. for (int i = size(nums) - 1; i >= 2; --i) {**

**7. if (i + 1 < size(nums) && nums[i] == nums[i + 1]) {**

**8. continue;**

**9. }**

**10. const auto& target = -nums[i];**

**11. int left = 0, right = i - 1;**

**12. while (left < right) {**

**13. if (nums[left] + nums[right] < target) {**

**14. ++left;**

**15. } else if (nums[left] + nums[right] > target) {**

**16. --right;**

**17. } else {**

**18. result.push\_back({nums[left], nums[right], nums[i]});**

**19. ++left; --right;**

**20. while (left < right && nums[left] == nums[left - 1]) {**

**21. ++left;**

**22. }**

**23. while (left < right && nums[right] == nums[right + 1]) {**

**24. --right;**

**25. }**