C++ Assignments | Bubble Sorting | Week 9

1. Which of the following(s) is/are true about bubble sort:
    1. It is stable sort
    2. It has a worst case space complexity of O(n)
    3. It involves swapping of adjacent elements
    4. After each iteration, the greatest element is placed at the end of the array.
2. What will the following array look like after one iteration of bubble sort [1,6,2,5,4,3].
    1. [1,3,2,4,5,6]
    2. [1,2,3,4,5,6]
    3. [1,2,5,4,3,6]
    4. [1,2,4,5,3,6]
3. In which case does bubble sort works in the most efficient way:
    1. When the array is sorted in increasing order
    2. When the array is sorted partially
    3. When the array is sorted in decreasing order.
    4. When the array is nearly sorted.
4. Sort the array in descending order using Bubble Sort.
5. Check if the given array is almost sorted. (elements are at-most one position away)

Note:- Please try to invest time doing the assignments which are necessary to build a strong foundation. Do not directly Copy Paste using Google or ChatGPT. Please use your brain😀.

1.It is stable sort:

2.It has a worst-case space complexity of O(n):

3.It involves swapping of adjacent elements:

- True. Bubble Sort repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
- 4.After each iteration, the greatest element is placed at the end of the array:
- True. After each full pass through the array, the largest unsorted element "bubbles" to its correct position at the end of the array.

Correct Option:3. `[1, 2, 5, 4, 3, 6]`

- 5. Sorted array in
- #include <iostream>
- using namespace std;

```
•
•   void bubbleSortDescending(int arr[], int n) {
•       for (int i = 0; i < n - 1; i++) {
•           for (int j = 0; j < n - i - 1; j++) {
•               if (arr[j] < arr[j + 1]) {
•                   swap(arr[j], arr[j + 1]);
•               }
•           }
•       }
•   }
•
•   int main() {
•       int arr[] = {1, 6, 2, 5, 4, 3};
•       int n = sizeof(arr) / sizeof(arr[0]);
•
•       bubbleSortDescending(arr, n);
•
•       cout << "Sorted array in descending order: ";
•       for (int i = 0; i < n; i++) {
•           cout << arr[i] << " ";
•       }
•       cout << endl;
•
•       return 0;
•   }
•   6.
•   #include <iostream>
•   using namespace std;
•
•   bool isAlmostSorted(int arr[], int n) {
•       for (int i = 0; i < n - 1; i++) {
•           if (arr[i] > arr[i + 1]) {
•               // Check if swapping arr[i] and arr[i+1] fixes the order
•               swap(arr[i], arr[i + 1]);
•               for (int j = 0; j < n - 1; j++) {
•                   if (arr[j] > arr[j + 1]) {
•                       return false;
•                   }
•               }
•               return true;
•           }
•       }
•       return true;
•   }
•
•   int main() {
•       int arr[] = {1, 2, 3, 5, 4, 6};
•       int n = sizeof(arr) / sizeof(arr[0]);
•
•       if (isAlmostSorted(arr, n)) {
•           cout << "The array is almost sorted." << endl;
•       } else {
```

```
            cout << "The array is not almost sorted." << endl;
        }

        return 0;
    }

```