Bubble Sorting Assignment - Week 9

1. Which of the following(s) is/are true about bubble sort?

- It is a stable sort: True
- It has a worst case space complexity of O(n): False (Space complexity is O(1))
- It involves swapping of adjacent elements: True
- After each iteration, the greatest element is placed at the end of the array: True

2. What will the following array look like after one iteration of bubble sort [1, 6, 2, 5, 4, 3]?

Answer: [1, 2, 5, 4, 3, 6]

3. In which case does bubble sort work in the most efficient way?

Answer: When the array is sorted in increasing order.
Bubble sort works most efficiently when the array is already sorted, achieving a time complexity of O(n).

4. Sort the array in descending order using Bubble Sort (C++ code):

cpp

```
#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] and arr[j+1]
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
      }
    }
  }
}
```

```
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] << " ";
   return 0;
}
```

5. Check if the given array is almost sorted (C++ code):

cpp

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

bool isAlmostSorted(int arr[], int n) {
   int misplacedCount = 0;
   for (int i = 0; i < n-1; i++) {
      if (arr[i] > arr[i+1]) {
         misplacedCount++;
      }
      if (misplacedCount > 1) {
         return false;
      }
   }
   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.";
   } else {
      cout << "The array is not almost sorted.";
   }
   return 0;
}
```