Problem Statement:

The aim of this project is to implement the **bubble sort algorithm** which is a well-known sorting technique used in data structures. It's called bubble sort or sinking sort because smaller values gradually "bubble" their way to the top of the array (i.e., toward the first element) like air bubbles rising in water, while the larger values sink to the bottom (end) of the array. The technique should use nested loops to make several passes through the array. Each pass compares successive pairs of elements. If a pair is in increasing order (or the values are equal), the bubble sort leaves the values as they are. If a pair is in decreasing order, the bubble sort swaps their values in the array. Consider the following array as an example, int $A = \{ 2, 6, 4, 8, 10, 12, 89, 68, 45, 37\}$; Once sorted using the bubble sort algorithm, it should have the following content: {2, 4, 6, 8, 10, 12, 37, 45, 68, 89}. The first pass should compare the first two element values of the array and swap them if necessary. It then should compare the second and third element values in the array. The end of this pass should compare the last two element values in the array and swap them if necessary. After one pass, the largest value will be in the last element. After two passes, the largest two values will be in the last two elements. In order to improve the performance of the bubble sort, we recommend following the recommendations given below for your final source code:

(*a*) After the first pass, the largest value is guaranteed to be in the highest-numbered element of the array; after the second pass, the two highest values are "in place"; and so on. Instead of making nine comparisons (for a 10-element array) on every pass, your algorithm should make only the eight necessary comparisons on the second pass, seven on the third pass, and so on.

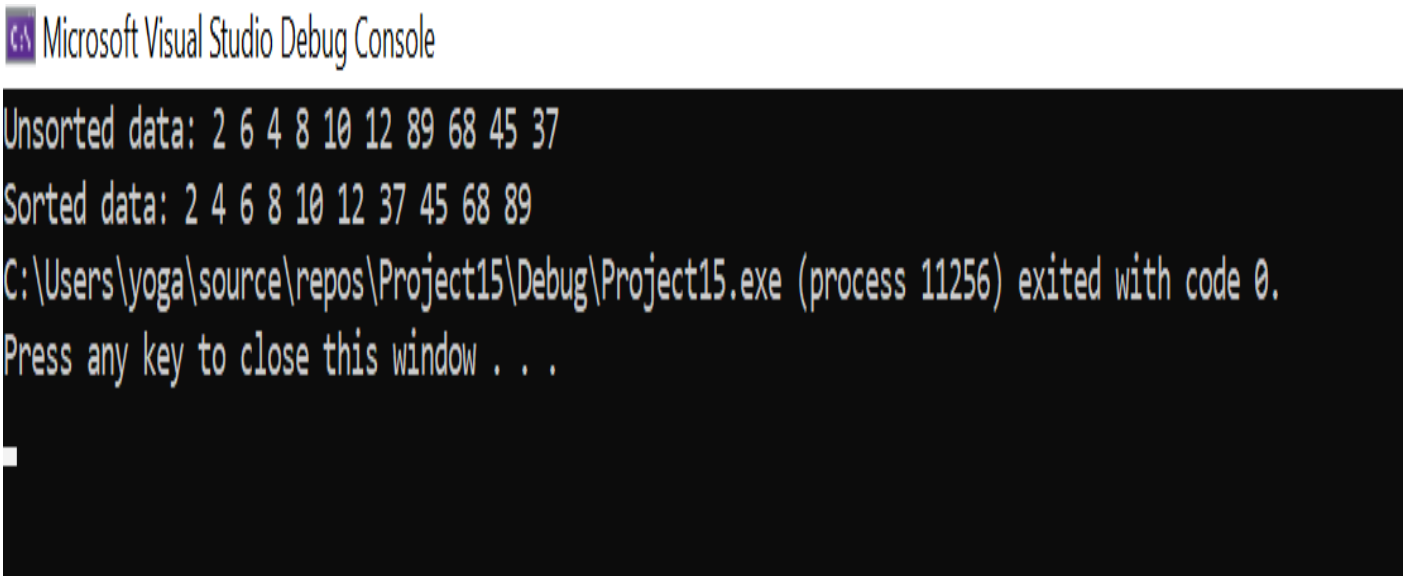
(*b*) The data in the array may already be in the proper order or near-proper order, so why make nine passes (of a 10-element array) if fewer will suffice? Modify the sort in (*a*) to check at the end of each pass whether any swaps have been made. If none have been made, the data must already be in the proper order, so the program should terminate. If swaps have been made, at least one more pass is needed.

By Muhammad Umar Farooq

Input:

```
#include <iostream>
using namespace std;
int main() {
    int swaps = 0;
    int dat[10] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
    cout << "Unsorted data: ";
    for (int z = 0; z <= 9; z++)
    {
        cout << dat[z] << " " ;
    }
    for (int i = 0; i < 9; i++)
    {
        if (i == 1 && swaps == 0)
        {
            break;
        }

        else
        {
            for (int j = 0; j < 9 - i; j++)
            {
                if (dat[j] > dat[j + 1])
                {
                    //Swapping algoritm
                    int temp = dat[j];
                    dat[j] = dat[j + 1];
                    dat[j + 1] = temp;
                    swaps++;
                }
            }

        }

    }
    cout << " " << endl;
    cout << "Sorted data: ";
    for (int z = 0; z <= 9; z++)
    {
        cout << dat[z] << " ";
    }

    return 0;
}
```

Output:


Microsoft Visual Studio Debug Console
Unsorted data: 2 6 4 8 10 12 89 68 45 37
Sorted data: 2 4 6 8 10 12 37 45 68 89
C:\Users\yoga\source\repos\Project15\Debug\Project15.exe (process 11256) exited with code 0.
Press any key to close this window . . .