Bubble sort and insertion sort

1. // arraySchools[5] = {4,2,5,1,2};

4. // Sorting algorithm - bubble
5. **for** (c = 0; c < n; c++) {
6. **for** (d = 0; d < (n-c-1); d++) {
8. **if** (arraySchools[d] > arraySchools[d+1]) {
10. t = arraySchools[d];
11. arraySchools[d] = arraySchools[d+1];
12. arraySchools[d+1] = t;
13. }
14. }
15. }


19. // Sorting algorithm - insertionSort
20. **for** (c = 0; c < n; c++) {
21. d = c;
22. **while** (d > 0 && arraySchools[d] < arraySchools[d-1]) {
24. t = arraySchools[d];
25. arraySchools[d] = arraySchools[d-1];
26. arraySchools[d-1] = t;
28. d--;
29. }
30. }

33. // OUTPUT: 1, 2, 2, 4, 5

More efficient bubble sort with midway break

1. #define MAX 5
2. **int** main(**void**){
3. **int** arraySchools[MAX] = {200,300,34,33,53};
4. **int** i,isNotSorted = 1,temp,numPasses = 0;
5. **for** (i = 0; (i < MAX - 1) && isNotSorted; i++){  //Created isNotSorted to break out if we find that there is no swaps done.
6. **int** j;
7. isNotSorted = 0;                // TO make it so that I will only change it once and not revert it in the same loop.
8. **for** (j = 0; j < MAX - i - 1; j++){
9. **if** (arraySchools[j] > arraySchools[j+1]){
10. temp = arraySchools[j];
11. arraySchools[j] = arraySchools[j+1];
12. arraySchools[j+1] = temp;
13. isNotSorted = 1;        //Swap detected. Will not break this loop.
14. }
15. }
16. numPasses++;        //Proof of concept. Try more complex arrays and an array that only requires one or two swaps.
17. }
18. printf("%d %d %d %d %d \n",arraySchools[0],arraySchools[1],arraySchools[2],arraySchools[3],arraySchools[4]);
19. printf("%d\n",numPasses);
20. **return** 0;
21. }
22. #define MAX 5
23. **int** main(**void**){
24. **int** arraySchools[MAX] = {32,2,34,59,11};
25. **int** i,smallest,temp;
26. **for** (i = 0; i < MAX - 1; i++){   // MAX - 1 as there is no point looking for the last position with one element left.
27. smallest = i;
28. **int** j;
29. **for** (j = i; j < MAX; j++){       // Searches for the position of the smallest element in the available array
30. **if** (arraySchools[smallest] > arraySchools[j]){
31. smallest = j;
32. }
33. }
34. // Swaps the smallest element with the i element that you are on.
35. temp = arraySchools[i];
36. arraySchools[i] = arraySchools[smallest];
37. arraySchools[smallest] = temp;
38. }
39. printf("%d %d %d %d %d \n",arraySchools[0],arraySchools[1],arraySchools[2],arraySchools[3],arraySchools[4]);
40. **return** 0;
41. }