Insertion Sort - Part 2

In *Insertion Sort Part 1*, you inserted one element into an array at its correct sorted position. Using the same approach repeatedly, can you sort an entire array?

Guideline: You already can place an element into a sorted array. How can you use that code to build up a sorted array, one element at a time? Note that in the first step, when you consider an array with just the first element, it is already sorted since there's nothing to compare it to.

In this challenge, print the array after each iteration of the insertion sort, i.e., whenever the next element has been inserted at its correct position. Since the array composed of just the first element is already sorted, begin printing after placing the second element.

Example.

$n = 7$
$arr = [3, 4, 7, 5, 6, 2, 1]$

Working from left to right, we get the following output:

```
3 4 7 5 6 2 1
3 4 7 5 6 2 1
3 4 5 7 6 2 1
3 4 5 6 7 2 1
2 3 4 5 6 7 1
1 2 3 4 5 6 7
```

Function Description

Complete the *insertionSort2* function in the editor below.

insertionSort2 has the following parameter(s):

- *int n:* the length of arr
- *int arr[n]:* an array of integers

Prints

At each iteration, print the array as space-separated integers on its own line.

Input Format

The first line contains an integer, n, the size of arr.
The next line contains n space-separated integers $arr[i]$.

Constraints

$1 \leq n \leq 1000$
$-10000 \leq arr[i] \leq 10000, 0 \leq i < n$

Output Format

Print the entire array on a new line at every iteration.

Sample Input

```
STDIN           Function
-----           --------
6               n = 6
1 4 3 5 6 2     arr = [1, 4, 3, 5, 6, 2]
```

Sample Output

```
1 4 3 5 6 2
1 3 4 5 6 2
1 3 4 5 6 2
1 3 4 5 6 2
1 2 3 4 5 6
```

Explanation

Skip testing 1 against itself at position 0. It is sorted.
Test position 1 against position 0: $4 > 1$, no more to check, no change.
Print arr
Test position 2 against positions 1 and 0:

- $3 < 4$, new position may be 1. Keep checking.
- $3 > 1$, so insert 3 at position 1 and move others to the right.

Print arr
Test position 3 against positions $2, 1, 0$ (as necessary): no change.
Print arr
Test position 4 against positions $3, 2, 1, 0$: no change.
Print arr
Test position 5 against positions $4, 3, 2, 1, 0$, insert 2 at position 1 and move others to the right.
Print arr

Contest ends in 2 hours

Submissions: 178
Max Score: 30
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

C++

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
```

```cpp
vector<string> split(const string &);

/*
 * Complete the 'insertionSort2' function below.
 *
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. INTEGER_ARRAY arr
 */

void insertionSort2(int n, vector<int> arr) {

}

int main()
{
    string n_temp;
    getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));

    string arr_temp_temp;
    getline(cin, arr_temp_temp);

    vector<string> arr_temp = split(rtrim(arr_temp_temp));

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {
        int arr_item = stoi(arr_temp[i]);

        arr[i] = arr_item;
    }

    insertionSort2(n, arr);

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}

```

Line: 1 Col: 1

Upload Code as File ☐ Test against custom input Run Code Submit Code

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |