Insertion Sort - Part 1

Sorting
One common task for computers is to sort data. For example, people might want to see all their files on a computer sorted by size. Since sorting is a simple problem with many different possible solutions, it is often used to introduce the study of algorithms.

Insertion Sort
These challenges will cover *Insertion Sort*, a simple and intuitive sorting algorithm. We will first start with a nearly sorted list.

Insert element into sorted list
Given a sorted list with an unsorted number e in the rightmost cell, can you write some simple code to insert e into the array so that it remains sorted?

Since this is a learning exercise, it won't be the most efficient way of performing the insertion. It will instead demonstrate the brute-force method in detail.

Assume you are given the array $arr = [1, 2, 4, 5, 3]$ indexed $0 \ldots 4$. Store the value of $arr[4]$. Now test lower index values successively from 3 to 0 until you reach a value that is lower than $arr[4]$, at $arr[1]$ in this case. Each time your test fails, copy the value at the lower index to the current index and print your array. When the next lower indexed value is smaller than $arr[4]$, insert the stored value at the current index and print the entire array.

Example
$n = 5$
$arr = [1, 2, 4, 5, 3]$
Start at the rightmost index. Store the value of $arr[4] = 3$. Compare this to each element to the left until a smaller value is reached. Here are the results as described:

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

Function Description

Complete the *insertionSort1* function in the editor below.

insertionSort1 has the following parameter(s):

- *n*: an integer, the size of arr
- *arr*: an array of integers to sort

Returns

- *None:* Print the interim and final arrays, each on a new line. No return value is expected.

Input Format

The first line contains the integer n, the size of the array arr.
The next line contains n space-separated integers $arr[0] \ldots arr[n-1]$.

Constraints

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

Output Format

Print the array as a row of space-separated integers each time there is a shift or insertion.

Sample Input

```
5
2 4 6 8 3
```

Sample Output

```
2 4 6 8 8
2 4 6 6 8
2 4 4 6 8
2 3 4 6 8
```

Explanation

3 is removed from the end of the array.
In the 1^{st} line $8 > 3$, so 8 is shifted one cell to the right.
In the 2^{nd} line $6 > 3$, so 6 is shifted one cell to the right.
In the 3^{rd} line $4 > 3$, so 4 is shifted one cell to the right.
In the 4^{th} line $2 < 3$, so 3 is placed at position 1.

Next Challenge

In the next Challenge, we will complete the insertion sort.

Contest ends in 3 days

Submissions: 0
Max Score: 50
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

C++

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

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'insertionSort1' function below.
 *
 * The function accepts following parameters:
```

```cpp
 *  1. INTEGER n
 *  2. INTEGER_ARRAY arr
 */

void insertionSort1(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;
    }

    insertionSort1(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 |