# Closest Numbers

locked



Submitted 21 minutes ago • Score: 25.00 Status: Accepted

| | | |
|---|---|---|
| ✔ Test Case #0 | ✔ Test Case #1 | ✔ Test Case #2 |
| ✔ Test Case #3 | ✔ Test Case #4 | ✔ Test Case #5 |

## Submitted Code

nguage: C++ Open in editor

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

using namespace std;

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

/*
 * Complete the 'closestNumbers' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

vector<int> closestNumbers(vector<int> arr) {
    //initializing vector
    vector<int> closestArr;
    // doing insertion sort for the vector
    for (int i = 1; i < (int)arr.size(); i++)
    {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
    // initialzing min with the difference of first two variables
    int min = arr[1]-arr[0];
    // itaerating through the elements of the vector
    for (int i=0; i<(int)arr.size()-1; i++){
        // checking whether the new difference is smaller than min
```

```
        if (arr[i+1]-arr[i] < min){
            closestArr = {arr[i], arr[i+1]};
            min = arr[i+1]-arr[i];

        } else if (arr[i+1]-arr[i] == min) {
            closestArr.push_back(arr[i]);
            closestArr.push_back(arr[i+1]);
        }
    }

    return closestArr;
}


int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    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;
    }

    vector<int> result = closestNumbers(arr);

    for (size_t i = 0; i < result.size(); i++) {
        fout << result[i];

        if (i != result.size() - 1) {
            fout << " ";
        }
    }

    fout << "\n";

    fout.close();

    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);
```

```
102
103     s.erase(
104         find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
105         s.end()
106     );
107
108     return s;
109 }
110
111 vector<string> split(const string &str) {
112     vector<string> tokens;
113
114     string::size_type start = 0;
115     string::size_type end = 0;
116
117     while ((end = str.find(" ", start)) != string::npos) {
118         tokens.push_back(str.substr(start, end - start));
119
120         start = end + 1;
121     }
122
123     tokens.push_back(str.substr(start));
124
125     return tokens;
126 }
127
```