Full Name:	Thota Ushaswini
Email:	ushaswinithota11@gmail.com
Test Name:	**Mock Test**
Taken On:	22 Aug 2025 12:22:10 IST
Time Taken:	19 min 3 sec/ 90 min
Invited by:	Ankush
Invited on:	22 Aug 2025 12:21:24 IST
Skills Score:	
Tags Score:	Algorithms 290/290 Arrays 95/95 Core CS 290/290 Data Structures 215/215 Easy 95/95 Medium 75/75 Queues 120/120 Search 75/75 Sorting 95/95 Strings 95/95 problem-solving 170/170

100%

290/290

scored in **Mock Test** in 19 min 3 sec on 22 Aug 2025 12:22:10 IST

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here -

	Question Description	Time Taken	Score	Status
Q1	Truck Tour > Coding	3 min 16 sec	120/ 120	
Q2	Pairs > Coding	11 min 3 sec	75/ 75	
Q3	Big Sorting > Coding	4 min 20 sec	95/ 95	

QUESTION 1

Truck Tour > Coding Algorithms Data Structures Queues Core CS

Needs Review

Score 120

QUESTION DESCRIPTION

Suppose there is a circle. There are N petrol pumps on that circle. Petrol pumps are numbered 0 to $(N-1)$ (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilometer for each litre of the petrol.

Input Format

The first line will contain the value of N.
The next N lines will contain a pair of integers each, i.e. the amount of petrol that petrol pump will give and the distance between that petrol pump and the next petrol pump.

Constraints:

$1 \leq N \leq 10^5$

$1 \leq \text{amount of petrol, distance} \leq 10^9$

Output Format

An integer which will be the smallest index of the petrol pump from which we can start the tour.

Sample Input

```
3
1 5
10 3
3 4
```

Sample Output

```
1
```

Explanation

We can start the tour from the second petrol pump.

CANDIDATE ANSWER

Language used: **C**

```c

/*
 * Complete the 'truckTour' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts 2D_INTEGER_ARRAY petrolpumps as parameter.
 */

int truckTour(int petrolpumps_rows, int petrolpumps_columns, int**
petrolpumps) {
    int start = 0;
    int surplus = 0;
    int deficit=0;
    for(int i =0;i<petrolpumps_rows;i++) {
        surplus += petrolpumps[i][0] - petrolpumps[i][1];
        if(surplus<0) {
            start = i+1;
```

```
            deficit += surplus;
            surplus = 0;
        }
    }
    if(surplus + deficit >= 0)
        return start;
    else
        return -1;

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0084 sec	7.25 KB
Testcase 2	Easy	Hidden case	Success	10	0.0084 sec	7.38 KB
Testcase 3	Easy	Hidden case	Success	10	0.0094 sec	7.13 KB
Testcase 4	Easy	Hidden case	Success	10	0.0119 sec	7.38 KB
Testcase 5	Easy	Hidden case	Success	10	0.0491 sec	17 KB
Testcase 6	Easy	Hidden case	Success	10	0.0354 sec	17 KB
Testcase 7	Easy	Hidden case	Success	10	0.0475 sec	17.1 KB
Testcase 8	Easy	Hidden case	Success	10	0.0341 sec	17 KB
Testcase 9	Easy	Hidden case	Success	10	0.0407 sec	17 KB
Testcase 10	Easy	Hidden case	Success	10	0.0409 sec	17.1 KB
Testcase 11	Easy	Hidden case	Success	10	0.0395 sec	17 KB
Testcase 12	Easy	Hidden case	Success	10	0.0475 sec	17.3 KB
Testcase 13	Easy	Hidden case	Success	10	0.0452 sec	17 KB

No Comments

QUESTION 2

Needs Review

Score 75

Pairs › Coding | Search | Algorithms | Medium | problem-solving | Core CS

QUESTION DESCRIPTION

Given an array of integers and a target value, determine the number of pairs of array elements that have a difference equal to the target value.

Example

$k = 1$

$arr = [1, 2, 3, 4]$

There are three values that differ by $k = 1$: $2 - 1 = 1$, $3 - 2 = 1$, and $4 - 3 = 1$. Return 3.

Function Description

Complete the *pairs* function below.

pairs has the following parameter(s):

- *int k:* an integer, the target difference
- *int arr[n]:* an array of integers

Returns

- *int:* the number of pairs that satisfy the criterion

Input Format

The first line contains two space-separated integers n and k, the size of arr and the target value.
The second line contains n space-separated integers of the array arr.

Constraints

- $2 \leq n \leq 10^5$
- $0 < k < 10^9$
- $0 < arr[i] < 2^{31} - 1$
- each integer $arr[i]$ will be unique

Sample Input

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

Sample Output

```
3
```

Explanation

There are 3 pairs of integers in the set with a difference of 2: [5,3], [4,2] and [3,1]. .

CANDIDATE ANSWER

Language used: **C**

```c

/*
 * Complete the 'pairs' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER k
 *  2. INTEGER_ARRAY arr
 */

int pairs(int k, int arr_count, int* arr) {
    int max_val = 0;
    int count = 0;
    for(int i = 0;i<arr_count;i++) {
        if(arr[i] > max_val)
            max_val = arr[i];
    }
    int *hash = (int*)calloc(max_val+2, sizeof(int));
    for(int i = 0;i < arr_count; i++) {
        hash[arr[i]] = 1;
    }
    for(int i =0;i<arr_count;i++) {
        if(arr[i] + k <= max_val && hash[arr[i] + k] == 1) {
            count++;
        }
    }
    free(hash);
    return count;

}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Hidden case	Success	5	0.012 sec	7.27 KB
Testcase 2	Easy	Hidden case	Success	5	0.0153 sec	8.13 KB
Testcase 3	Easy	Hidden case	Success	5	0.0089 sec	8.13 KB
Testcase 4	Easy	Hidden case	Success	5	0.0098 sec	7.38 KB
Testcase 5	Easy	Hidden case	Success	5	0.0125 sec	9 KB
Testcase 6	Easy	Hidden case	Success	5	0.0366 sec	29.9 KB
Testcase 7	Easy	Hidden case	Success	5	0.0261 sec	31.3 KB
Testcase 8	Easy	Hidden case	Success	5	0.0172 sec	18.3 KB
Testcase 9	Easy	Hidden case	Success	5	0.0243 sec	27.1 KB
Testcase 10	Easy	Hidden case	Success	5	0.0577 sec	43 KB
Testcase 11	Easy	Hidden case	Success	5	0.1732 sec	246 KB
Testcase 12	Easy	Hidden case	Success	5	0.1847 sec	246 KB
Testcase 13	Easy	Hidden case	Success	5	0.178 sec	248 KB
Testcase 14	Easy	Hidden case	Success	5	0.1763 sec	246 KB
Testcase 15	Easy	Hidden case	Success	5	0.1663 sec	247 KB
Testcase 16	Easy	Sample case	Success	0	0.0097 sec	7.25 KB
Testcase 17	Easy	Sample case	Segmentation Fault	0	0.0439 sec	6.63 KB
Testcase 18	Easy	Sample case	Success	0	0.0083 sec	7.25 KB

No Comments

QUESTION 3

Needs Review

Score 95

Big Sorting › Coding

Sorting | Strings | Algorithms | Easy | Data Structures | Arrays | problem-solving | Core CS

QUESTION DESCRIPTION

Consider an array of numeric strings where each string is a positive number with anywhere from 1 to 10^6 digits. Sort the array's elements in *non-decreasing*, or ascending order of their integer values and return the sorted array.

Example

$unsorted = ['1', '200', '150', '3']$

Return the array ['1', '3', '150', '200'].

Function Description

Complete the *bigSorting* function in the editor below.

bigSorting has the following parameter(s):

- *string unsorted[n]:* an unsorted array of integers as strings

Returns

- *string[n]:* the array sorted in numerical order

Input Format

The first line contains an integer, n, the number of strings in $unsorted$.
Each of the n subsequent lines contains an integer string, $unsorted[i]$.

Constraints

- $1 \leq n \leq 2 \times 10^5$
- Each string is guaranteed to represent a positive integer.
- There will be no leading zeros.
- The total number of digits across all strings in $unsorted$ is between 1 and 10^6 (inclusive).

Sample Input 0

```
6
314159265358979323846264338327 95
1
3
10
3
5
```

Sample Output 0

```
1
3
3
5
10
314159265358979323846264338327 95
```

Explanation 0

The initial array of strings is $unsorted = [314159265358979323846264338327 95, 1, 3, 10, 3, 5]$. When we order each string by the real-world integer value it represents, we get:

$$1 \leq 3 \leq 3 \leq 5 \leq 10 \leq 314159265358979323846264338327 95$$

We then print each value on a new line, from smallest to largest.

Sample Input 1

```
8
1
2
100
123034798498573417183401923 71
3084193741082937
3084193741082938
111
200
```

Sample Output 1

```
1
2
100
111
200
3084193741082937
3084193741082938
123034798498573417183401923 71
```

CANDIDATE ANSWER

Language used: **C**

```c

/*
 * Complete the 'bigSorting' function below.
 *
 * The function is expected to return a STRING_ARRAY.
 * The function accepts STRING_ARRAY unsorted as parameter.
 */

/*
 * To return the string array from the function, you should:
 *     - Store the size of the array to be returned in the result_count
variable
 *     - Allocate the array statically or dynamically
 *
 * For example,
 * char** return_string_array_using_static_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     static char* a[5] = {"static", "allocation", "of", "string", "array"};
 *
 *     return a;
 * }
 *
 * char** return_string_array_using_dynamic_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     char** a = malloc(5 * sizeof(char*));
 *
 *     for (int i = 0; i < 5; i++) {
 *         *(a + i) = malloc(20 * sizeof(char));
 *     }
 *
 *     *(a + 0) = "dynamic";
 *     *(a + 1) = "allocation";
 *     *(a + 2) = "of";
 *     *(a + 3) = "string";
 *     *(a + 4) = "array";
 *
 *     return a;
 * }
 *
 */
 int bigStrCmp(const void* a, const void* b) {
    char* sa = *(char**)a;
    char* sb = *(char**)b;
    int la = strlen(sa);
    int lb = strlen(sb);
    if(la != lb)
        return la - lb;
    return strcmp(sa, sb);
 }
char** bigSorting(int unsorted_count, char** unsorted, int* result_count) {
    qsort(unsorted, unsorted_count, sizeof(char*), bigStrCmp);
    *result_count = unsorted_count;
    return unsorted;

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED

Testcase 1	Easy	Sample case	Success	0	0.0073 sec	7.25 KB
Testcase 2	Medium	Hidden case	Success	10	0.0074 sec	7.13 KB
Testcase 3	Medium	Hidden case	Success	10	0.0137 sec	7.88 KB
Testcase 4	Hard	Hidden case	Success	15	0.0563 sec	8.25 KB
Testcase 5	Hard	Hidden case	Success	15	0.0193 sec	8.25 KB
Testcase 6	Hard	Hidden case	Success	15	0.0116 sec	8 KB
Testcase 7	Hard	Hidden case	Success	15	0.0243 sec	9.46 KB
Testcase 8	Hard	Hidden case	Success	15	0.1296 sec	15.5 KB
Testcase 9	Easy	Sample case	Success	0	0.0077 sec	7.25 KB

No Comments

PDF generated at: 22 Aug 2025 07:13:27 UTC