Mock Test > subiyamehek2003@gmail.com

Full Name:	Subiya Mehek
Email:	subiyamehek2003@gmail.com
Test Name:	**Mock Test**
Taken On:	19 Aug 2025 20:40:22 IST
Time Taken:	30 min 53 sec/ 90 min
Invited by:	Ankush
Invited on:	19 Aug 2025 20:40:01 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 30 min 53 sec on 19 Aug 2025 20:40:22 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**	10 min 19 sec	120/ 120	⚠
Q2	Pairs > **Coding**	7 min	75/ 75	⚠
Q3	Big Sorting > **Coding**	10 min 3 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) {
    long long total_petrol = 0, total_distance = 0;
    long long tank = 0;
    int start = 0;

    for (int i = 0; i < petrolpumps_rows; i++) {
        long long petrol = petrolpumps[i][0];
        long long distance = petrolpumps[i][1];
        total_petrol += petrol;
        total_distance += distance;

        tank += petrol - distance;

        if (tank < 0) {
            // Can't reach next pump, reset start
            start = i + 1;
            tank = 0;
        }
    }

    // If total petrol is less than total distance, no solution
    if (total_petrol < total_distance) return -1;

    return start;
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.0088 sec	7.38 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.0082 sec	7.25 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.0106 sec	7.38 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.0082 sec	7.25 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.044 sec	17.1 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.039 sec	16.9 KB
Testcase 7	Easy	Hidden case	✓ Success	10	0.0457 sec	17.1 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.05 sec	17.3 KB
Testcase 9	Easy	Hidden case	✓ Success	10	0.0404 sec	17.1 KB
Testcase 10	Easy	Hidden case	✓ Success	10	0.036 sec	17.1 KB
Testcase 11	Easy	Hidden case	✓ Success	10	0.0373 sec	17.1 KB
Testcase 12	Easy	Hidden case	✓ Success	10	0.0386 sec	16.9 KB
Testcase 13	Easy	Hidden case	✓ Success	10	0.0342 sec	17.1 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
```

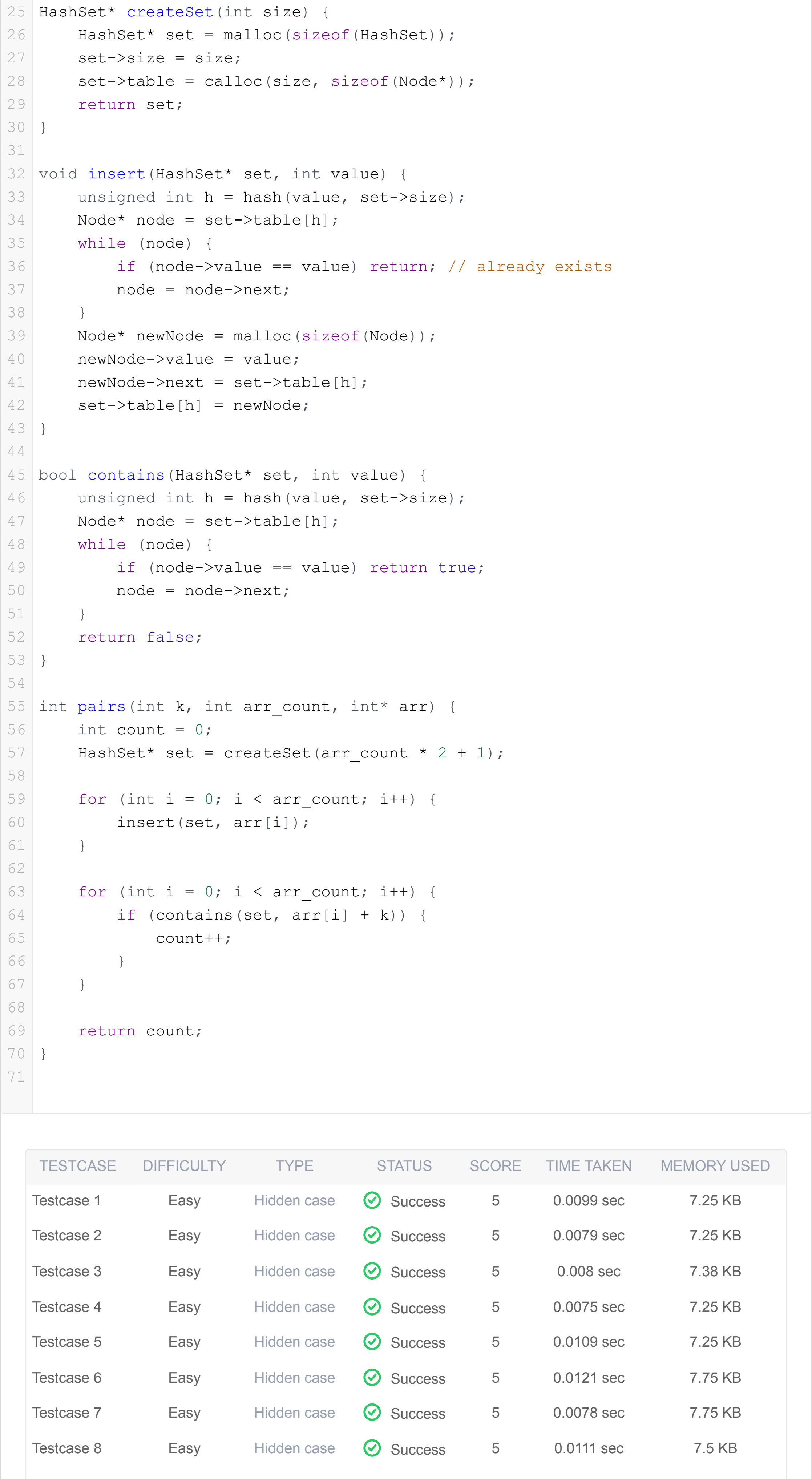
Explanation

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
 */

typedef struct Node {
    int value;
    struct Node* next;
} Node;

typedef struct HashSet {
    Node** table;
    int size;
} HashSet;

unsigned int hash(int key, int size) {
    return ((unsigned int)key) % size;
}

```



```c
HashSet* createSet(int size) {
    HashSet* set = malloc(sizeof(HashSet));
    set->size = size;
    set->table = calloc(size, sizeof(Node*));
    return set;
}

void insert(HashSet* set, int value) {
    unsigned int h = hash(value, set->size);
    Node* node = set->table[h];
    while (node) {
        if (node->value == value) return; // already exists
        node = node->next;
    }
    Node* newNode = malloc(sizeof(Node));
    newNode->value = value;
    newNode->next = set->table[h];
    set->table[h] = newNode;
}

bool contains(HashSet* set, int value) {
    unsigned int h = hash(value, set->size);
    Node* node = set->table[h];
    while (node) {
        if (node->value == value) return true;
        node = node->next;
    }
    return false;
}

int pairs(int k, int arr_count, int* arr) {
    int count = 0;
    HashSet* set = createSet(arr_count * 2 + 1);

    for (int i = 0; i < arr_count; i++) {
        insert(set, arr[i]);
    }

    for (int i = 0; i < arr_count; i++) {
        if (contains(set, arr[i] + k)) {
            count++;
        }
    }

    return count;
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Hidden case	✓ Success	5	0.0099 sec	7.25 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.0079 sec	7.25 KB
Testcase 3	Easy	Hidden case	✓ Success	5	0.008 sec	7.38 KB
Testcase 4	Easy	Hidden case	✓ Success	5	0.0075 sec	7.25 KB
Testcase 5	Easy	Hidden case	✓ Success	5	0.0109 sec	7.25 KB
Testcase 6	Easy	Hidden case	✓ Success	5	0.0121 sec	7.75 KB
Testcase 7	Easy	Hidden case	✓ Success	5	0.0078 sec	7.75 KB
Testcase 8	Easy	Hidden case	✓ Success	5	0.0111 sec	7.5 KB

Testcase 9	Easy	Hidden case	✓ Success	5	0.0147 sec	7.5 KB
Testcase 10	Easy	Hidden case	✓ Success	5	0.0123 sec	7.63 KB
Testcase 11	Easy	Hidden case	✓ Success	5	0.0259 sec	13.8 KB
Testcase 12	Easy	Hidden case	✓ Success	5	0.0417 sec	13.6 KB
Testcase 13	Easy	Hidden case	✓ Success	5	0.0483 sec	13.8 KB
Testcase 14	Easy	Hidden case	✓ Success	5	0.0331 sec	14 KB
Testcase 15	Easy	Hidden case	✓ Success	5	0.066 sec	13.8 KB
Testcase 16	Easy	Sample case	✓ Success	0	0.0077 sec	7.38 KB
Testcase 17	Easy	Sample case	✓ Success	0	0.0075 sec	7.38 KB
Testcase 18	Easy	Sample case	✓ Success	0	0.0109 sec	7.5 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

$unsorted = [\text{'1', '200', '150', '3'}]$

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
31415926535897932384626433832795
1
3
10
```

```
3
5
```

Sample Output 0

```
1
3
3
5
10
31415926535897932384626433832795
```

Explanation 0

The initial array of strings is $unsorted = [31415926535897932384626433832795, 1, 3, 10, 3, 5]$. When we order each string by the real-world integer value it represents, we get:

$$1 \leq 3 \leq 3 \leq 5 \leq 10 \leq 31415926535897932384626433832795$$

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

Sample Input 1

```
8
1
2
100
12303479849857341718340192371
3084193741082937
3084193741082938
111
200
```

Sample Output 1

```
1
2
100
111
200
3084193741082937
3084193741082938
12303479849857341718340192371
```

CANDIDATE ANSWER

Language used: **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
```

```c
 *
 * 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 compareNumbers(const void* a, const void* b) {
    char* num1 = *(char**)a;
    char* num2 = *(char**)b;

    int len1 = strlen(num1);
    int len2 = strlen(num2);

    if (len1 != len2)
        return len1 - len2;    // shorter number is smaller

    return strcmp(num1, num2); // lexicographic compare if equal length
}

char** bigSorting(int unsorted_count, char** unsorted, int* result_count) {
    qsort(unsorted, unsorted_count, sizeof(char*), compareNumbers);
    *result_count = unsorted_count;
    return unsorted;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.0075 sec	7.38 KB
Testcase 2	Medium	Hidden case	✓ Success	10	0.0121 sec	7.25 KB
Testcase 3	Medium	Hidden case	✓ Success	10	0.0172 sec	7.63 KB
Testcase 4	Hard	Hidden case	✓ Success	15	0.0183 sec	8.38 KB
Testcase 5	Hard	Hidden case	✓ Success	15	0.0319 sec	8.63 KB
Testcase 6	Hard	Hidden case	✓ Success	15	0.0096 sec	8.25 KB
Testcase 7	Hard	Hidden case	✓ Success	15	0.0452 sec	9.39 KB
Testcase 8	Hard	Hidden case	✓ Success	15	0.1327 sec	15.8 KB

Testcase 9	Easy	Sample case	✓	Success	0	0.0075 sec	7.13 KB

No Comments

PDF generated at: 19 Aug 2025 15:58:38 UTC