Questions

1. [Double on Match](#DoubleonMatch) [CANDIDATE ANSWER](#CANDIDATE_ANSWER_DoubleOnMatch)
2. [Team Formation 2](#TeamFormation2)  (Medium Algorithms Problem Solving) [CANDIDATE ANSWER](#CANDIDATEANSWERTeamFormation2)
3. [Distance Between Two Points](#DistanceBetweenTwoPoints) [CANDIDATE ANSWER](#CANDIDATEANSWER_distancebetween2points)
4. Value of Properties Owned (Database)
5. [Sprint Training](#SprintTraining) ( Data Structures Medium Algorithms Arrays Problem Solving)

**Double on Match**

**QUESTION DESCRIPTION**

Given an array of long integers (arr) and a number (num). Iterate through the elements in arr and double the value of num whenever an element equals num. arr can be reordered before the iteration to maximize the value of num. Find the maximum possible value of num.

Example

arr = [1, 2, 4, 11, 12, 8]

num = 2

Iterating through arr:

|  |  |
| --- | --- |
| arr | num |
|  | 2 |
| 1 | 2 |
| 2 | 4 |
| 4 | 8 |
| 11 | 8 |
| 12 | 8 |
| 8 | 16 |

The maximal value of num = 16. Note that arr could have been reordered before iterating.

**Function Description**

Complete the function doubleSize in the editor below.

doubleSize has the following parameter(s):

long int arr[n]: an array of long integers

long int num: the base long integer

**Returns:**

long int: the maximal value of num

Constraints

* 1 ≤ n ≤ 10
* 0 ≤ arr[i] ≤ 10
* 0 ≤ num ≤ 10

**Input Format for Custom Testing**

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the size of the array arr.

Each of the next n lines contains an integer arr[i] where 0 ≤ i < n.

The last line contains a long integer, num.

**Sample Case 0**

**Sample Input 0**

STDIN Function

----- --------

5 → arr[ ] size n = 5

1 → arr = [1, 2, 3, 1, 2]

2

3

1

2

1 → num = 1

**Sample Output 0**

4

**Explanation 0**

Rearrange arr to arr = {1, 1, 2, 2, 3}.

|  |  |
| --- | --- |
| arr | num |
|  | 1 |
| 1 | 2 |
| 1 | 2 |
| 2 | 4 |
| 2 | 4 |
| 3 | 4 |

**Sample Case 1**

**Sample Input 1**

STDIN Function

3 → arr[ ] size n = 3

1 → arr = [1, 1, 1]

1

1

1 → num = 1

**Sample Output 1**

2

**Explanation 1**

|  |  |
| --- | --- |
| arr | num |
|  | 1 |
| 1 | 2 |
| 1 | 2 |
| 1 | 2 |

Sample Case 2

**Sample Input 2**

STDIN Function

5 → arr[ ] size n = 5

2 → arr = [2, 5, 4, 6, 8]

5

4

6

8

2 → num = 2

**Sample Output 2**

16

**Explanation 2**

Rearrange arr to arr = {2, 4, 5, 6, 8}

|  |  |
| --- | --- |
| arr | num |
|  | 2 |
| 2 | 4 |
| 4 | 8 |
| 5 | 8 |
| 6 | 8 |
| 8 | 16 |

**Hint 1**

Imagine that the optimal ordering of the array results in doubling the value of b some certain

number of times, for example, it doubles it 5 times. Think about what exact values in the array

double the value of b while iterating over the array.

**Answer:** Those values will be b, b\*2, b\*4, b\*8, b\*16 and so on.

**Hint 2**

Is there any ordering of the array that guarantees that the values b, b\*2, b\*4, b\*16, ... are

processed in this exact order so we have as many doubles as possible?

**Answer:** Yes, ordering the array in a non-descending order guarantees that.

**CANDIDATE ANSWER**

**Language used: Java 8**

class Result {

/\*

\* Complete the 'doubleSize' function below.

\*

\* The function is expected to return a LONG\_INTEGER.

\* The function accepts following parameters:

\* 1. LONG\_INTEGER\_ARRAY arr

\* 2. LONG\_INTEGER b

\*/

public static long doubleSize(List<Long> arr, long b) {

// Write your code here

}

}

**Team Formation 2**

**QUESTION DESCRIPTION**

FC Codelona is trying to assemble a team from a roster of available players. They have a minimum

number of players they want to sign, and each player needs to have a skill rating within a certain range.

Given a list of players' skill levels with desired upper and lower bounds, determine how many teams can be

created from the list.

**Example**

skills = [12, 4, 6, 13, 5, 10]

minPlayers = 3

minLevel = 4

maxLevel = 10

* The list includes players with skill levels [12, 4, 6, 13, 5, 10].
* They want to hire at least 3 players with skill levels between 4 and 10, inclusive.
* Four of the players with the following skill levels { 4, 6, 5,10} meet the criteria.
* There are 5 ways to form a team of 3 players : {4, 5, 6}, {4, 6, 10}, {4, 5,10}, {5, 6, 10}, and {4, 5, 6, 10}.
* Return 5.

**Function Description**

Complete the function countTeams in the editor below.

countTeams has the following parameter(s):

int skills[n]: an array of integers that represent the skill level per player

int minPlayers: the minimum number of team members required

int minLevel: the lower limit for skill level, inclusive

int maxLevel: the upper limit for skill level, inclusive

**Return**

int: the total number of teams that can be formed per the criteria

**Constraints**

* 1 ≤ n ≤ 20
* 1 ≤ minPlayers ≤ n
* 1 ≤ minPlayers ≤ n
* 1 ≤ minLevel ≤ maxLevel ≤ 1000
* 1 ≤ skills[i] ≤ 1000

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

1. The first line contains an integer n, the size of the array skills.
2. The next n lines each contain an element skills[i] where 0 ≤ i < n.
3. The next line contains an integer, minPlayers, the minimum number of players to be included in the team.
4. The next line contains an integer, minLevel, the lower limit of skill level to select
5. The next line contains an integer, maxLevel, the upper limit of skill level to select

**Sample Case 0**

**Sample Input 0**

STDIN Function

4 → skills[ ] size n = 4

4 → skills = [4, 8, 5, 6]

8

5

6

1 → minPlayers = 1

5 → minLevel = 5

7 → maxLevel = 7

**Sample Output 0**

3

**Explanation 0**

* The list includes players with skill levels [4, 8, 5, 6].
* They want to hire at least 1 player with skill levels between 5 and 7, inclusive.
* Two of the players with the following skill levels { 5, 6} meet the criteria
* There are 3 ways to form a team of at least 1 player : {5}, {6}, {5, 6} .
* Returns 3.

**Sample Case 1**

**Sample Input 1**

STDIN Function

**4 → skills[ ] size n = 4**

**4 → skills = [4, 8, 5, 6]**

**8**

**5**

**6**

**2 → minPlayers = 2**

**5 → minLevel = 5**

**7 → maxLevel = 7**

**Sample Output 1**

1

**Explanation 1**

* The list includes players with skill levels [4, 8, 5, 6].
* They want to hire at least 2 players with skill levels between 5 and 7, inclusive.
* Two of the players with the following skill levels { 5, 6} meet the criteria
* There is only one ways to form a team of at least 2 players : {5, 6} .
* Returns 1

**Sample Case 2**

**Sample Input 2**

STDIN Function

4 → skills[ ] size n = 4

4 → skills = [4, 8, 5, 6]

8

5

6

2 → minPlayers = 2

7 → minLevel = 7

8 → maxLevel = 8

**Sample Output 2**

0

**Explanation 2**

* The list includes players with skill levels [4, 8, 5, 6].
* They want to hire at least 2 players with skill levels between 7 and 8, inclusive.
* One of the players with the following skill levels { 8 } meet the criteria.
* There is no way to form a team of at least 2 players.
* Returns 0.

**Hint 1**

* Count the number of eligible players. Now you just need to count the number of ways you can choose at least minPlayers from the eligible players. Look over the constraint of n.

**Hint 2**

* We can say that the number of ways to choose at least minPlayers from the eligible players is equal to number of ways to choose exactly p players such that p > minPlayers.

CANDIDATE ANSWER

Language used: Java 8

class Result {

/\*

\* Complete the 'countTeams' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts following parameters:

\* 1. INTEGER\_ARRAY skills

\* 2. INTEGER minPlayers

\* 3. INTEGER minLevel

\* 4. INTEGER maxLevel

\*/

public static int countTeams(List<Integer> skills, int minPlayers, int minLevel, int maxLevel) {

// Write your code here

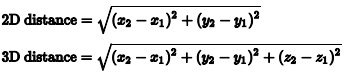
}

}

**Distance Between Two Points**

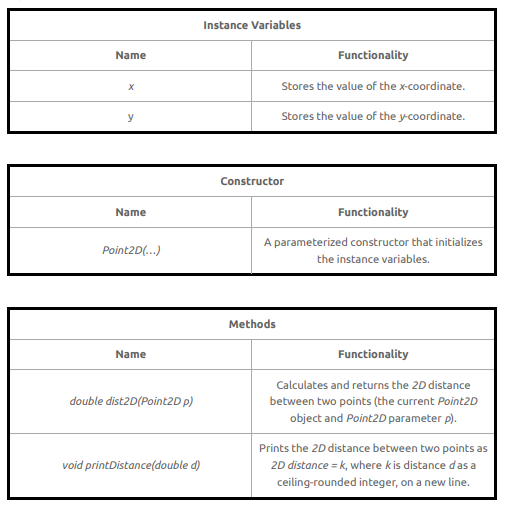
QUESTION DESCRIPTION

This challenge involves points in two and three dimensional space. The classes and methods to implement will store values for coordinates as well as calculate distances between points. The 2D and 3D distances between two points are calculated using the following formulae:

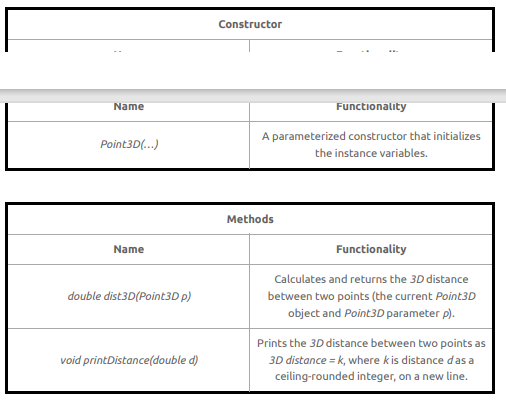


Implement the classes and methods defined below:

1. A superclass named Point2D:



1. A derived class named Point3D that extends Point2D



A main method is provided in the locked portion of the editor. It parses six values representing point coordinates and calls the implemented constructors and methods. Here, x[1], y[1], and z[1] represent the coordinates of the first point, and x[2], y[2], and z[2] represent the coordinates of the second point.

Note that printed output must exactly match the above for the test cases to pass.

**Constraints**

* -128 ≤ x,y,z ≤ 127

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

Line: description

1. x[1]: integer

2. y[1]: integer

3. z[1]: integer

4. x[2]: integer

5. y[2]: integer

6. z[2]: integer

**Sample Case 0**

**Sample Input 0**

STDIN Function

1 → x[1] = 1

2 → y[1] = 2

3 → z[1] = 3

4 → x[2] = 4

5 → y[2] = 5

6 → z[2] = 6

**Sample Output 0**

2D distance = 5

3D distance = 6

**Explanation 0**

* For the first point: x = 1, y = 2, z = 3.
* For the second point: x = 4, y = 5, z = 6.
* The formula gives a 2D distance of sqrt(3 + 3 ) = 4.242640687119285. The ceiling is 5.
* The formula gives a 3D distance of sqrt(3 + 3 + 3 ) = 5.196152422706632. The ceiling is 6.

**Sample Case 1**

**Sample Input 1**

STDIN Function

-1 → x[1] = -1

-2 → y[1] = -2

4 → z[1] = 4

-7 → x[2] = -7

-1 → y[2] = -1

-1 → z[2] = -1

**Sample Output 1**

2D distance = 7

3D distance = 8

**Explanation 1**

* For the first point, x = -1, y = -2, z = 4
* For the second point, x = -7, y = -1, z = -1
* The differences are x: -6, y: 1, z: -5
* The squares of differences are x: 36, y: 1, z: 25
* 2D distance calculation: sqrt(36+1) = 6.08276253, ceiling = 7
* 3D distance calculation: sqrt(36+1+25) = 7.874007874, ceiling = 8

**CANDIDATE ANSWER**

Language used: Java 8

// Write your class implementations here. Do not use access modifiers when declaring your classes.

// Write your code here

**Value of Properties Owned (Database)**

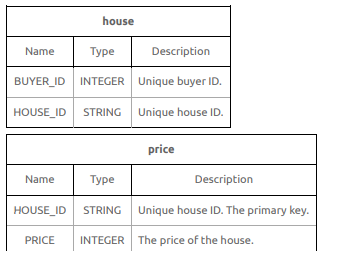
QUESTION DESCRIPTION

There are two tables in a database of real estate owners. One has ownership information and the other has price information, in millions. An owner may own multiple houses, but a house will have only one owner.

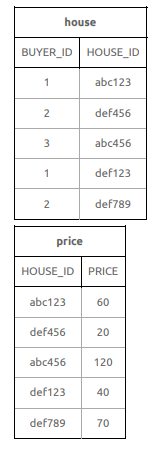
Write a query to print the IDs of the owners who have at least 100 million worth of houses and own more than 1 house. The order of output does not matter. The result should be in the format: BUYER\_ID TOTAL\_WORTH

Schema

There are 2 tables: house , price .



Sample Data Tables



**Sample Output**

1 100

Explanation

* 1 has a total of (60 + 40) = 100 million worth houses and is included in the results.
* 3 has a total of 120 million worth houses but has only 1 house.
* 2 has a total of (20 + 70) = 90 million worth of houses.

CANDIDATE ANSWER

Language used: MySQL

/\*

Enter your query below.

Please append a semicolon ";" at the end of the query

\*/

**Sprint Training**

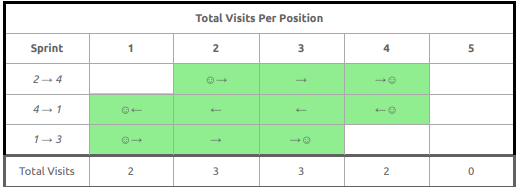
**QUESTION DESCRIPTION**Pat is an ordinary kid who works hard to be a great runner. As part of training, Pat must run sprints of different intervals on a straight trail. The trail has numbered markers that the coach uses as goals. Pat's coach provides a list of goals to reach in order. Each time Pat starts at, stops at, or passes a marker it is considered a visit. Determine the lowest numbered marker that is visited the most times during Pat's day of training.

**Example**

n = 5

sprints = [2, 4, 1, 3]

if the number of markers on the trail, n = 5, and assigned sprints = [2, 4, 1, 3], Pat first sprints from position 2 → 4. The next sprint is from position 4 → 1, and then 1 → 3. A marker numbered position p is considered to be visited each time Pat either starts or ends a sprint there and each time it is passed while sprinting. The total number of visits to each position in the example is calculated like so:



Pat has visited markers 2 and 3 a total of 3 times each. Since 2 < 3, the lowest numbered marker that is Visited the most times during Pat's day of training is 2.

**Function Description**

Complete the function getMostVisited in the editor below.

getMostVisited has the following parameter(s):

* int n: an integer denoting the number of markers along the trail
* int sprints[m]: an array of integers denoting the sequence of markers to reach, beginning at the marker

shown in sprints[0].

**Returns:**

int: an integer denoting Pat's most visited position on the trail after performing all m − 1 sprints. If there are multiple such answers, return the smallest one.

**Constraints**

* 1 ≤ n ≤ 10
* 2 ≤ m ≤ 10
* 1 ≤ sprints[i] ≤ m (where 0 ≤ i < m)
* sprints[i-1] ≠ sprints[i] (where 0 < i < m)

**Input Format for Custom Testing**

* Input from stdin will be processed as follows and passed to the function.
* The first line contains an integer n, the number of markers along the path.
* The second line contains an integer m, the number of markers in the list of goals.
* The next m lines each contain an element sprints[i] where 0 ≤ i < m.

**Sample Case 0**

**Sample Input 0**

STDIN Function Parameters

10 → n = 10

4 → sprints[ ] size m = 4

1 → sprints = [1, 5, 10, 3]

5

10

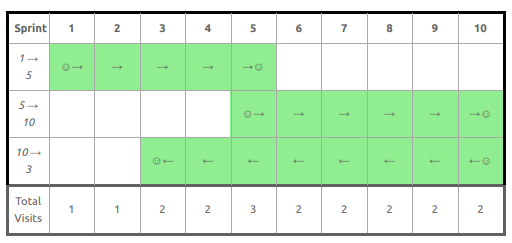
3

**Sample Output 0**

**5**

**Explanation 0**

**Given sprints = [1, 5, 10, 3], Pat performs the following sequence of sprints:**



In the table above, Pat visited marker 5 the most.

Sample Case 1

**Sample Input 1**

STDIN Function Parameters

5 → n = 5

2 → sprints[ ] size m = 2

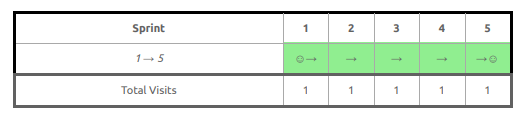
1 → sprints = [1, 5]

5

**Sample Output 1**

1

**Explanation 1**

Given sprints = [1, 5], Pat performs the following sprint:

In the table above, every marker is visited the same number of times. Return the smallest of these, which is 1.

**Sample Case 2**

**Sample Input 2**

STDIN Function Parameters

9 → n = 9

4 → sprints[ ] size m = 4

9 → sprints = [9, 7, 3, 1]

7

3

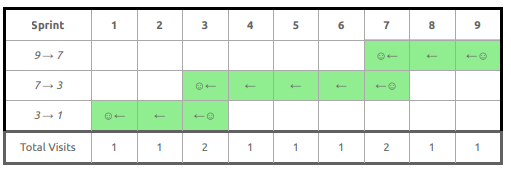
1

**Sample Output 2**

3

**Explanation 2**

Given sprints = [9, 7, 3, 1], Pat performs the following sequence of sprints:



In the table above, Pat visited positions 3 and 7 the most. Return the smallest of these, which is 3.

**CANDIDATE ANSWER**

Language used: Java 8

class Result {

/\*

\* Complete the 'getMostVisited' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts following parameters:

\* 1. INTEGER n

\* 2. INTEGER\_ARRAY sprints

\*/

public static int getMostVisited(int n, List<Integer> sprints) {

// Write your code here}

}

file:///D:/Jas\_STG/Training/jas\_training/HackerRank%20Questions/Assessment/Shinas/Report\_Software\_Engineer\_Skills\_assessment\_-\_STG\_Infotech\_India\_LLP\_ping2sowmi\_outlook.com.pdf