








































Practice Arena

Practice problems aimed to improve your coding skills.

-  PRACTICE-02_SCAN-PRINT
-  PRACTICE-03_TYPES
-  LAB-PRAC-02_SCAN-PRINT
-  LAB-PRAC-01
-  PRACTICE-04_COND
-  BONUS-PRAC-02
-  LAB-PRAC-03_TYPES
-  PRACTICE-05_COND-LOOPS
-  LAB-PRAC-04_COND
-  LAB-PRAC-05_CONDLLOOPS
-  PRACTICE-07_LOOPS-ARR
-  LAB-PRAC-06_LOOPS
-  LAB-PRAC-07_LOOPS-ARR
-  LABEXAM-PRAC-01_MIDSEM
-  PRACTICE-09_PTR-MAT
-  LAB-PRAC-08_ARR-STR
-  PRACTICE-10_MAT-FUN
-  LAB-PRAC-09_PTR-MAT
-  LAB-PRAC-10_MAT-FUN
-  PRACTICE-11_FUN-PTR
-  LAB-PRAC-11_FUN-PTR
-  LAB-PRAC-12_FUN-STRUC
-  LABEXAM-PRAC-02_ENDSEM
-  LAB-PRAC-13_STRUC-NUM
 -  Too tired to create a story - part I
 -  Too tired to create a story - part II
 -  Too tired to create a story - part III
 -  Point Proximity
 -  The Bisection Method
 -  The pace is too fast
 -  A Question on Quadrilaterals
 -  The Trapezoidal Technique
 -  Constrained Candy Crush
 -  Major Mobile Madness
 -  The Newton Raphson Method
 -  The Palindrome Decomposition
-  LAB-PRAC-14_SORT-MISC

Point Proximity

LAB-PRAC-13_STRUC-NUM

Point Proximity [20 marks]

Problem Statement

The first line of the input will give three strictly positive integers n , k , q , separated by a space. In the next n lines, we will give you a list of k -dimensional point. On each of the n lines we will give you the k integer coordinates of a point and a code name corresponding to that point. All points will be k -dimensional and the code name will be a string with no spaces and at most 99 characters. q will be an integer between 1 and n (both included).

Look at the q -th point in the list (i.e. if $q = 1$, look at the first point in the list and if $q = 2$ then look at the 2nd point in the list). Call this the *query point*. Your job is to find the point in the list that is farthest from the query point in terms of squared Euclidean distance. In your output, you will have two lines. In the first line, print the code name of the point farthest from the query point and in the second line, print the squared Euclidean distance between the query point and the point farthest from it. If there are multiple farthest points in the list, print the code name of the point that appears first on the list.

We can represent a k -dimensional point as an array. Given two k -dimensional points as two integer arrays (remember coordinates of the points are all integers) a and b , the squared Euclidean distance between the two is calculated as

$$sqEuclid(a, b) = \sum_{i=0}^{k-1} (a[i] - b[i])^2$$

Caution

1. All code names will be unique i.e. no two lines will have the same code name.
 2. q will be strictly positive i.e. it will look like a human-readable position in the list that starts from 1, not an array index that starts from 0.
 3. Coordinates of the points can be negative integers too.
 4. Since your coordinates are integers, the squared Euclidean distances (in particular the maximum squared Euclidean distance) will be an integer too.
 5. Be careful about extra/missing lines and extra/missing spaces in your output.
-

EXAMPLE:

INPUT

6 3 1

1 1 1 IITKANPUR

11 2 3 IITMADRAS

-1 10 1 IITBOMBAY

-10 2 1 IITDELHI

5 -5 5 IITKHARAGPUR

-10 2 1 IITD

OUTPUT:
IITDELHI
122

Explanation: Both IITDELHI and IITD are equally far from IITKANPUR but IITDELHI appears first on the list.

Grading Scheme:

Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

 **Start Solving!** (/editor/practice/6258)