Member Count: 595,545 - October 27, 2013 [Get Time]

[TOPCODER]®

Login

Now

Competitions

Overview

Copilot Opportunities

Design

Development

UI Development

QA and Maintenance

CloudSpokes

Algorithm

Single Round Matches (SRM)

Overview

Track Information

Statistics

Match Archive

Match Overviews

Match Winners

Match Results

Match Editorials

Problem Archive

Recent Color Changes

Data Feeds

Launch Arena

Marathon Match

High School

The Digital Run

Submit & Review

TopCoder Networks

Events

Statistics

Tutorials

Forums

Surveys

My TopCoder

Help Center

About TopCoder

Member Search:

Statistics Problem Statement

Problem Statement for FlowerGarden

Problem Statement

You are planting a flower garden with bulbs to give you joyous flowers throughout the year. However, you wish to plant the flowers such that they do not block other flowers while they are visible.

You will be given a int[] **height**, a int[] **bloom**, and a int[] **wilt**. Each type of flower is represented by the element at the same index of **height**, **bloom**, and **wilt**. **height** represents how high each type of flower grows, **bloom** represents the morning that each type of flower springs from the ground, and **wilt** represents the evening that each type of flower shrivels up and dies. Each element in **bloom** and **wilt** will be a number between 1 and 365 inclusive, and **wilt**[*i*] will always be greater than **bloom**[*i*]. You must plant all of the flowers of the same type in a single row for appearance, and you also want to have the tallest flowers as far forward as possible. However, if a flower type is taller than another type, and both types can be out of the ground at the same time, the shorter flower must be planted in front of the taller flower to prevent blocking. A flower blooms in the morning, and wilts in the evening, so even if one flower is blooming on the same day another flower is wilting, one can block the other.

You should return a int[] which contains the elements of **height** in the order you should plant your flowers to acheive the above goals. The front of the garden is represented by the first element in your return value, and is where you view the garden from. The elements of **height** will all be unique, so there will always be a well-defined ordering.

Definition

Class:	FlowerGarden
Method:	getOrdering
Parameters:	int[], int[], int[]
Returns:	int[]
Method signature:	int[] getOrdering(int[] height, int[] bloom, int[] wilt)

(be sure your method is public)

Constraints

- **height** will have between 2 and 50 elements,

Member Search:
Handle: Go
Advanced Search

inclusive.

- **bloom** will have the same number of elements as **height**
- **wilt** will have the same number of elements as **height**
- **height** will have no repeated elements.
- Each element of **height** will be between 1 and 1000, inclusive.
- Each element of **bloom** will be between 1 and 365, inclusive.
- Each element of **wilt** will be between 1 and 365, inclusive.
- For each element *i* of **bloom** and **wilt**, **wilt**[*i*] will be greater than **bloom**[*i*].

Examples

0)

```
{5,4,3,2,1}
{1,1,1,1,1}
{365,365,365,365,365}
```

```
Returns: { 1,  2,  3,  4,  5 }
```

These flowers all bloom on January 1st and wilt on December 31st. Since they all may block each other, you must order them from shortest to tallest.

1)

```
{5,4,3,2,1}
{1,5,10,15,20}
{4,9,14,19,24}
```

```
Returns: { 5,  4,  3,  2,  1 }
```

The same set of flowers now bloom all at separate times. Since they will never block each other, you can order them from tallest to shortest to get the tallest ones as far forward as possible.

2)

```
{5,4,3,2,1}
{1,5,10,15,20}
{5,10,15,20,25}
```

```
Returns: { 1,  2,  3,  4,  5 }
```

Although each flower only blocks at most one other, they all must be ordered from shortest to tallest to prevent any blocking from occurring.

3)

```
{5,4,3,2,1}
{1,5,10,15,20}
{5,10,14,20,25}
```

```
Returns: { 3,  4,  5,  1,  2 }
```

The difference here is that the third type of flower wilts one day earlier than the blooming

of the fourth flower. Therefore, we can put the flowers of height 3 first, then the flowers of height 4, then height 5, and finally the flowers of height 1 and 2. Note that we could have also ordered them with height 1 first, but this does not result in the maximum possible height being first in the garden.

4)

```
{1,2,3,4,5,6}
{1,3,1,3,1,3}
{2,4,2,4,2,4}
```

```
Returns: { 2,  4,  6,  1,  3,  5 }
```

5)

```
{3,2,5,4}
{1,2,11,10}
{4,3,12,13}
```

```
Returns: { 4,  5,  2,  3 }
```

This problem was used for:
2004 TCCC Online Round 1 - Division I, Level Two