Problem Statement

Find the least-cost 3D box that can hold two given 3D boxes. The boxes must be kept orthogonal (not tilted) although they can be rotated at any multiple of 90 degrees in 3-space. Cost is the total area of cardboard required for the enclosing box, where four sides have one layer of cardboard and two ends have two layers of cardboard. The two boxes that are enclosed may not overlap (consider them already filled). You are given the dimensions of the two boxes as int[]s, where each int[] contains exactly three elements.

Definition

Class:	ShipBoxes
Method:	bestCost
Parameters:	int[], int[]
Returns:	int
Method signature:	int bestCost(int[] box1, int[] box2)
(be sure your method is public)	

Constraints

- **box1** and **box2** must each contain exactly 3 elements.
- Each element of **box1** and **box2** will be between 1 and 10,000, inclusive.

Examples

0)

```
{1,4,9}
{1,4,9}
Returns: 140
```

These two boxes have the shape of the monolith from 2001, A Space Odyssey---mostly flat. The most economical way to ship them is to stack them on top of each other and put the flaps at the small ends. The top of the box then is 4x9, the sides are 2x9, and the ends are 2x4; the total area of cardboard required is thus 140.

1)

```
{1,1,1}
{1,1,1}
Returns: 12
```

2)

```
{1,9000,9000}
{40,40,40}
Returns: 164214000
```

3)

```
{6570,6076,5880}
{7595,3,1835}
Returns: 324635290
```