

Given a 6×6 2D Array, *arr*:

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

We define an hourglass in *A* to be a subset of values with indices falling in this pattern in *arr*'s graphical representation:

```
a b c
  d
e f g
```

There are **16** hourglasses in *arr*, and an *hourglass sum* is the sum of an hourglass' values. Calculate the hourglass sum for every hourglass in *arr*, then print the *maximum* hourglass sum.

For example, given the 2D array:

```
-9 -9 -9  1 1 1
 0 -9  0  4 3 2
-9 -9 -9  1 2 3
 0  0  8  6 6 0
 0  0  0 -2 0 0
 0  0  1  2 4 0
```

We calculate the following **16** hourglass values:

```
-63, -34, -9, 12,
-10,  0, 28, 23,
-27, -11, -2, 10,
 9, 17, 25, 18
```

Our highest hourglass value is **28** from the hourglass:

```
 0 4 3
   1
 8 6 6
```

Note: If you have already solved the Java domain's *Java 2D Array* challenge, you may wish to skip this challenge.

Function Description

Complete the function *hourglassSum* in the editor below. It should return an integer, the maximum hourglass sum in the array.

hourglassSum has the following parameter(s):

- arr*: an array of integers

Input Format

Each of the **6** lines of inputs *arr[i]* contains **6** space-separated integers *arr[i][j]*.

Constraints

- $-9 \leq arr[i][j] \leq 9$
- $0 \leq i, j \leq 5$

Output Format

Print the largest (maximum) hourglass sum found in *arr*.

Sample Input

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Sample Output

Explanation

arr contains the following hourglasses:

1	1	1	1	1	0	1	0	0	0	0	0
	1			0			0			0	
1	1	1	1	1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
	1			1			0			0	
0	0	2	0	2	4	2	4	4	4	4	0
1	1	1	1	1	0	1	0	0	0	0	0
	0			2			4			4	
0	0	0	0	0	2	0	2	0	2	0	0
0	0	2	0	2	4	2	4	4	4	4	0
	0			0			2			0	
0	0	1	0	1	2	1	2	4	2	4	0

The hourglass with the maximum sum (**19**) is:

2 4 4
2
1 2 4