ASDAN China 阿思丹

ACSL Tiles

Time Remaining 2:03 mins:s

Time is almost up! Remember, only your Saved answers are recorded when the time runs out.

CLASS / SOURCE NAME
Name your class acslsr

PROBLEM: ACSL Tiles is a one-person game played with rectangular tiles. Each tile has a single-digit number, 0-9, at each end. At the start of the game, there are 4 rows, each with a single-digit number from 0-9; thus the number 405 represents the starting numbers 0, 4, 0, 5. Tiles can be re-oriented; thus, the tiles 04 and 40 are the same tile.

Tiles, called the *hand*, are given to the player, and other tiles are put into a *draw pile*. The initial number of tiles in your *hand* and in the *draw pile* varies from game to game. The goal of the game is to build rows by placing a tile from your *hand* at the right end of a row whose last number matches one of the numbers on the tile. If no tiles in your *hand* can be placed, tiles are added to the end of your *hand* from the *draw pile*, until a tile from the *draw pile* can be placed.

More specifically, on each turn, try to match your tiles starting with the first one in your *hand*. See if any can be added to one of the rows, starting with the row after the one where the last tile was placed, rotating back to Row 1 if necessary. Start looking at Row 1 when the game starts. However, if the last tile placed was a *double* (i.e., both numbers are the same), another tile must be placed on that row before any other match can be considered. If you cannot place any tiles in your *hand*, add tiles from the *draw pile* to your *hand* until a tile can be placed.

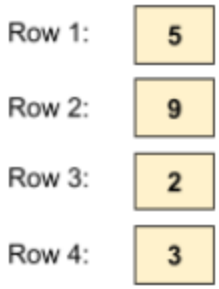
The game ends when you've placed all tiles in your *hand* OR you cannot place any of the tiles in your *hand* and you've exhausted the *draw pile*. At that point, find the sum of the single-digit numbers that are still in your hand.

EXAMPLE:

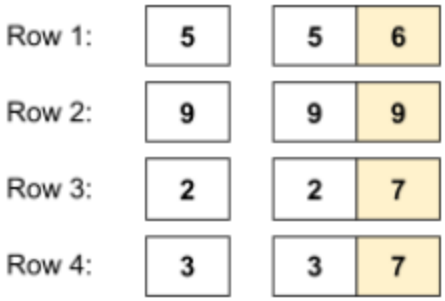
Input	Output
5923 56 27 73 34 99 45 32 17 64 57 18 11 36 92 22 50 82	16

Explanation:

The game starts with 4 rows having numbers 5, 9, 2, 3.

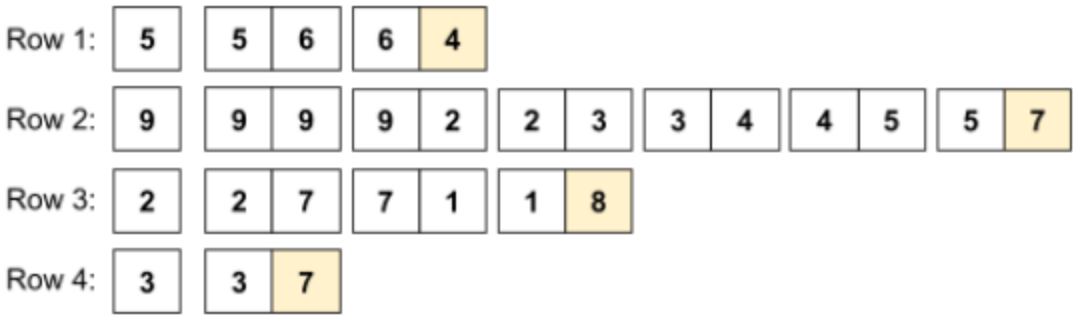


The tile 56 is placed on Row 1; the tile 27 is placed on Row 3; the tile 73 is placed on Row 4 (note that it's rotated, so that the 3 matches); the tile 34 is kept in your hand; and the tile 99 is placed on Row 2.

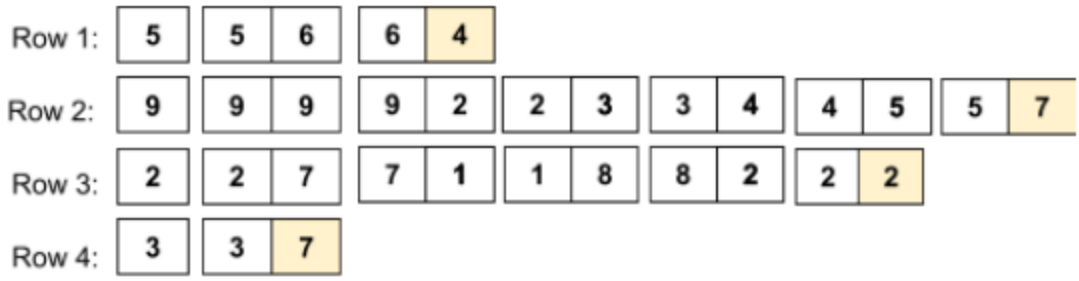


The 99 is a double, so the next tile to be placed must match a 9. None of the tiles in your hand match, so 36 is added to your hand from the draw pile. That tile cannot be placed, so it stays in your hand. Next the 92 is drawn; it can be placed on Row 2. Your hand is now 34 45 32 17 64 57 18 11 36 and the draw pile is 22 50 82.

The following moves are made: 32 on Row 2 (rotated), 34 on Row 2, 45 on Row 2, 17 on Row 3 (rotated), 64 on Row 1, 57 on Row 2, 18 on Row 3. Here is what the board looks like:



The tiles left in your hand are 11 and 36; neither can be placed, so draw 22 (it cannot be placed), then 50 (it cannot be placed), and then 82. The 82 can be placed on Row 3. Your hand is now 11 36 22 50 and the draw pile is empty. The 22 tile can be placed and then the game is over. The final board is as follows:



Your hand is 11 36 50, so the sum of the single-digit numbers on those tiles is 1+1+3+6+5+0 =16.

INPUT: Input an integer, between 0 and 9999 inclusive, that gives the initial numbers from Row 1 to Row 4. It is followed by two strings of numbers, from 0 to 99 inclusive, each separated by a single space. The first string has the tiles in your hand; the second string has the tiles in the draw pile. The numbers in each string represent the two numbers on each tile; a single-digit number indicates that at least one of the numbers on the tile is a 0.

OUTPUT: After placing the tiles using the rules above, output the sum of the single-digit numbers on the tiles left in your hand. If you've placed all of your tiles, then you'll output a 0.

SAMPLE INPUT	SAMPLE OUTPUT
5923 56 27 73 34 99 45 32 17 64 57 18 11 36 92 22 50 82	16
1324 85 31 32 96 25 1 68 30 35 42 11 78 39 19 9 81	0
7836 57 62 19 97 3 11 28 92 66 87 45 68 55 58 98 38 14 53 88 44 94 81 76 74 99 27 20	102
4 50 0 39 98 2 99 63 46 92 74 14 58 68 33 37 51 42 95 60 67 77 84 7 96 8 35 10 19 22 11 82 40	16

8937 63 84 6 57 8 2 30 9 87 52 5 58 40 62 54 27 96 35 99 61 56 14 51 88 13	71

问题：ACSL Tiles 是一个与长方形图块有关的单人游戏。每个图块的两端都有一个位于 0~9 之间的个位数。游戏开始时，共 4 行，每一行都包含一个位于 0~9 之间的个位数；也就是说，数字 405 就代表起始数字为0、4、0、5。图块可以改变方向；例如，图块 04 和图块 40 相同。

玩家收到的图块堆称为 *hand*, 其他的图块都放在 *draw pile 中。不同游戏*中，*hand* 和 *draw pile* 中图块的初始数字不同。游戏目标是构建行，将 *hand* 中的一个图块放在行的右端，使得这行的最后一个数字与图块上其中一个数字匹配。如果 *hand* 中没有可放置的图块，*draw pile 中的图块将会在 hand* 末尾进行补位，直到补位图块可放在行中时停止补位。

具体来说，每一轮游戏都要从 *hand 中的第一个图块开始尝试与行的最后一位数进行匹配。 从上一块已放置图块所在行之后的行开始，思考某一图块是否可以被放在其中一行中。如需要，可循环返回第* 1 *行开始。游戏开始时，从第* 1 *行开始观察。如果行的最后一个图块 是 double（*即图块上两位数字相同），那么必须在考虑其他行的匹配之前先在该行末尾放置图块。如果 *hand* 中没有可放置的图块，*draw pile 中图块将会在 hand* 末尾进行补位，直到补位的图块可放在行中时停止补位。

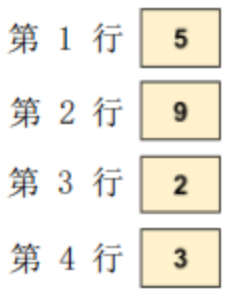
当 *hand* 中所有图块都已放置 ***或*** *hand* 中没有图块可放置且 *draw pile* 中图块被用尽时，游戏结束。此时，求 *hand* 中剩余图块上个位数的总和。

示例:

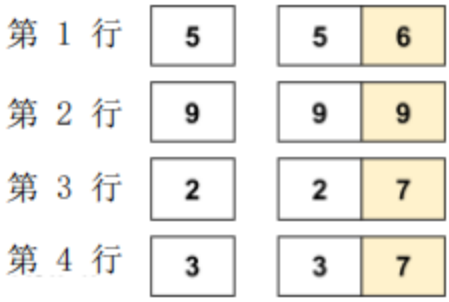
输入	输出
5923 56 27 73 34 99 45 32 17 64 57 18 11 36 92 22 50 82	16

详解:

游戏开始时，4 行初始数字为 5、9、2、3。

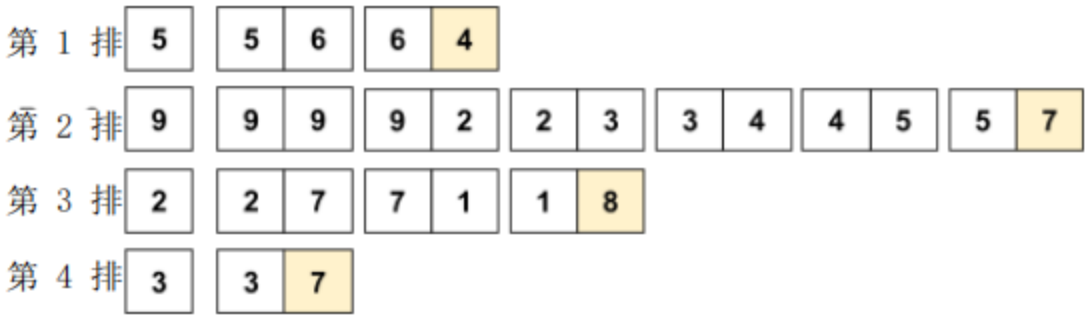


图块 56 放在第 1 行；图块 27 放在第 3 行；图块 73 放在第 4 行(注意：图块经过旋转，与 3 匹配）；图块 34 留在 *hand 中*；图块99 放在第 2 行。

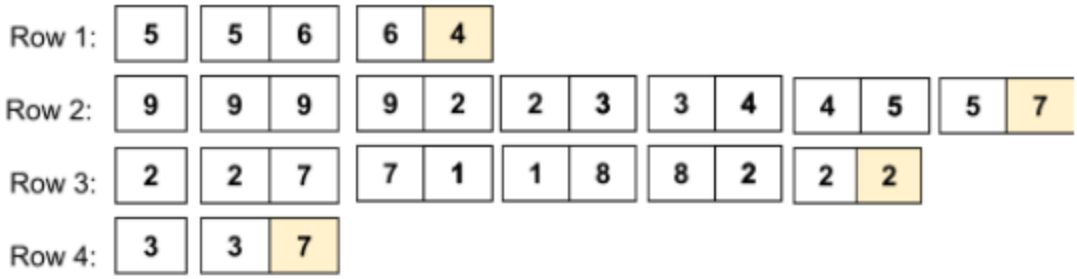


图块 99 是两位相同数字，因此下一个被放置的图块必须与 9 匹配。但是由于 *hand* 中没有这样的图块，*draw pile* 中图块 36 补位。36 与 9 不匹配，滞留在 *hand* 中。接下来图块 92 补位，与 9 匹配可以放在第 2 行。现在，*hand* 中图块包括 34 45 32 17 64 57 18 11 36；*draw pile* 中图块包括 22 50 82。

接下来：图块 32 放在第 2 行（经过旋转），图块 34 放在第 2 行, 图块 45 放在第 2 行，图块 17 放在第 3 行（经过旋转）, 图块 64 放在第 1 行，图块 57 放在第 2 行，图块 18 放在第 3 行。情况如下：



此时 hand 中剩下的图块只有 11 和 36；都不能放置，所以 *draw pile* 中图块 22 （不能放置）、 50 （不能放置）和 82 补位。图块 82 可以放在第 3 行。*hand* 中现在剩下 11、36、22、50 且此时*draw pile* 中图块用尽。随后可放置图块 22 ， 游戏结束。最终情况如下：



此时 hand 中剩下图块11 36 50, 其中个位数字相加可得：1+1+3+6+5+0 =16。

输入: 输入一个 0 - 9999 之间（包括 0和9999 ）的整数，这个整数中的各个数字为第 1行到第 4 行的初始数字。初始数字后面跟着两串 0 - 99 之间（包括 0 和 99）的数字, 数字之间用空格隔开。第一行数字为 *hand* 中图块上的数

字；第 2 行数字为 *draw pile* 中图块上的数字。每一串中的数字都代表各图块上的两个数字；如果只有单独一个个位数意味着图块上的数字中至少有一位是 0。

输出: 按照以上规则放置图块后，输出 *hand* 中剩余图块上个位数的总和。如果 *hand 中* 没有剩余图块，则输出 0。

样本输入	样本输出
5923 56 27 73 34 99 45 32 17 64 57 18 11 36 92 22 50 82	16
1324 85 31 32 96 25 1 68 30 35 42 11 78 39 19 9 81	0
7836 57 62 19 97 3 11 28 92 66 87 45 68 55 58 98 38 14 53 88 44 94 81 76 74 99 27 20	102
4 50 0 39 98 2 99 63 46 92 74 14 58 68 33 37 51 42 95 60 67 77 84 7 96 8 35 10 19 22 11 82 40	16
8937 63 84 6 57 8 2 30 9 87 52 5 58 40 62 54 27 96 35 99 61 56 14 51 88 13	71

Compiler Python 3.10/CPython

```
37      if elements[element_id][0] == end:
38          rst = (elements[element_id][0], elements[element_id][1])
39          rows[row].append(rst)
```

```
45          rows[row].append(rst)
46          last_touched_row = row
47          elements[element_id] = -1
48          return True
49      return False
50
51  def match_element_to_rows(element_id):
52      '''
53      return successful or not
```

Submit Use your own custom input

Previous submissions

Click file name to reuse submission. NOTE: this will overwrite existing text.

Date File name Score Status Compiler