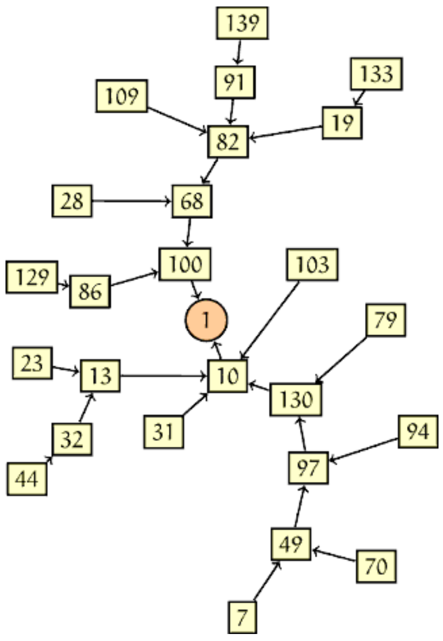
In this challenge, you have to implement an algorithm to establish if a given positive integer `num` is a Happy number, and how many steps of the algorithm are needed to establish it.

You have to repeatedly transform the given `num` into the sum of its squared digits:

- If after the transformation the new number is equal to `1`, `num` is a Happy number and the algorithm stops.
- If after the transformation, the new number is not equal to `1`, you have to transform it again.



If a number can't be reduced to `1`, sooner or later the algorithm will enter into an infinitely repeating loop:

```
... 20, 4, 16, 37, 58, 89, 145, 42, 20, 4 ...
```

Given a positive integer `num`, implement a function that returns:

- If `num` is a Happy number, the string `"HAPPY x"` with the "**x**" being the number of steps necessary to reduce `num` to 1.
- If `num` is not a Happy number, the string `"SAD x"` with the "**x**" being the number of steps necessary to enter into the infinite loop reaching so any number in the series `4, 16, 37, 58, 89, 145, 42, 20`, plus the number of steps necessary to obtain again that number.

Look at the examples below for a better visualization.

Examples

```
happyAlgorithm(139) ➞ "HAPPY 5"

// Step 1: Transform 139
// 1² + 3² + 9² = 1 + 9 + 81 = 91

// Step 2: Transform 91
// 9² + 1² = 81 + 1 = 82

// Step 3: Transform 82
// 8² + 2² = 64 + 4 = 68

// Step 4: Transform 68
// 6² + 8² = 36 + 64 = 100

// Step 5: Transform 100
// 1² + 0² + 0² = 1 + 0 + 0 = 1

// The algorithm stops at step 5: 139 is a Happy number

happyAlgorithm(67) ➞ "SAD 10"

// Step 1: Transform 67
// 6² + 7² = 36 + 49 = 85

// Step 2: Transform 85
// 8² + 5² = 64 + 25 = 89
// It entered into the infinite loop...
// ...but we have to demonstrate that is a loop!

// Step 3: Transform 89
// 8² + 9² = 64 + 81 = 145

// Step 4: Transform 145: result is 42
// Step 5: Transform 42: result is 20
// Step 6: Transform 20: result is 4
// Step 7: Transform 4: result is 16
// Step 8: Transform 16: result is 37
// Step 9: Transform 37: result is 58
// Step 10: Transform 58: result is 89

// 89 was the result of step 2: it's a loop
// The algorithm stops at step 10: 67 is not a Happy number

happyAlgorithm(1) ➞ "HAPPY 1"

// Step 1: Transform 1
// 1² = 1

// The algorithm stops at step 1: 1 is a Happy number

happyAlgorithm(89) ➞ "SAD 8"

// Step 1: Transform 89: result is 145
// Step 2: Transform 145: result is 42
// Step 3: Transform 42: result is 20
// Step 4: Transform 20: result is 4
// Step 5: Transform 4: result is 16
// Step 6: Transform 16: result is 37
// Step 7: Transform 37: result is 58
// Step 8: Transform 58: result is 89

// 89 was the original number: it's a loop
// The algorithm stops at step 8: 89 is not a Happy number
```

Notes

- The transformation of a single-digit number is, trivially, the square of the digit (see example #3).
- If the given number is `1`, a step is needed to establish if it's Happy (see example #3).
- To establish if a number is not happy, **you have to find the number of steps necessary to obtain again a number already found** (it can be a number obtained through a transformation as in example #2 or the same given number as in example #4).
- You can expect only positive integers as input, without exceptions to handle.