Wiffle Ball, Baseball, and Softball

You are an operational manager working at a community baseball field that allows people of all ages to play 3 variations of ball games: 1) wiffle ball, 2) baseball, and 3) softball. All 3 variations of the game use different balls. To help the community get the proper ball they need to play their game quickly, you set up 3 different bins behind the ball park where each bin is only meant to hold one type of ball. In other words, one bin is meant to hold all of the baseballs, another bin to hold all of the wiffle balls, and another bin to hold all of the softballs. You do not care which bin holds which type of ball. You only care that every bin holds only 1 type of ball.

Unfortunately, at the end of every day you find that people lazily place the balls in any bin they please, leading to all of the bins holding all different types of balls.

Being the great operational manager that you are, you spend time at the end of every day sorting the balls into bins again. You do not want to spend all night doing this, so your goal is to minimize the number of balls you have to move to ensure each bin only has 1 type of ball in it. In other words, how can you move the least number of balls to get each bin holding only 1 type of ball?

You may assume the following:

- Each move consists of moving one ball from one bin to another (i.e. one ball at a time)
- The only problem is to minimize the number of movements between the bins.
- Each bin has infinite capacity.
- The total number of balls will never exceed 2^31.

The Input

The input is a line containing 9 integers, where the first 3 represent the number of wiffle balls, baseballs, and softballs in bin 1 (respectively), the next 3 represent the number of wiffle balls, baseballs, and softballs in bin 2 (respectively), and likewise the next 3 represent the number of wiffle balls, baseballs, and softballs in bin 3 (respectively).

<u>Example Input</u>: *15 8 31 30 12 8 10 15 20*

<u>In Words</u>: *15 wiffle balls in bin 1, 8 baseballs in bin 1, 31 softballs in bin 1, 30 wiffle balls in bin 2, 12 baseballs in bin 2…….*

The Output

The output of your program will be 3 uppercase letters 'W', 'B', 'S' (representing the types "wiffle ball", "baseball", and "softball") in order of which bin contains the type followed by a space and then an integer representing the minimum number of ball movements that was needed. If there is more than one order of bins that produces the same minimum number of ball movements, then the first string in alphabetical order representing the minimal configuration should be printed (i.e., string BWS > string WSB)..

<u>Example Output (for example input):</u> *SWB 73*

<u>In Words</u>: *bin 1 has soft balls, bin 2 has wiffle balls, and bin 3 has baseballs… and it took a minimal 73 movements of balls to achieve this separation.*

Cycles of Sequence

Consider this algorithm:

```
1.  input n
2.  print n
3.  if n = 1 then STOP
4.          if n is odd then:  n ← 3n + 1
5.          else: n ← n/2
6.  GOTO 2
```

Therefore, if *n* is given as 10, the following sequence of numbers would be printed: 10 5 16 8 4 2 1

It is believed that this sequence will terminate (with a 1 being the last number printed) and this has been verified to be true for all integers *n* such that $0 < n < 1,000,000$ (although it's actually true for many more numbers than this).

The number of numbers printed (including the 1) for a given *n* is referred to as the cycle-length of *n*. In our example, the cycle length of 10 is 7.

For any two numbers *x* and *y* you are to determine the maximum cycle length of all numbers between *x* and *y.*

You may assume the following:

- No operation overflows a 32-bit integer

The Input
The values *x* and *y* separated by a space.

Example Input: *1 10*

The Output
An integer that represents the maximum cycle length.

Example Output (for example input): *20*