Counting Change

On 16 Mar 2017, the Monetary Authority of Singapore proposed a new legal tender limit for coins[1]! Previously, a payee is obliged to accept up to 40 pieces of 5-cent coins, 20 pieces of 10-cent coins, 10 pieces of 20-cent coins, 20 pieces of 50-cent coins and any number of one-dollar coins in a single transaction.

However, under the new proposed legal tender limit for coins, a payee is only obliged to accept up to 10 pieces of any particular type of coin in a single transaction. For example, if someone buys a 90-cent item, he can choose to pay with 10 pieces of 5-cent coins and 4 pieces of 10-cent coins, but he cannot pay with 18 pieces of 5-cent coins since the payee is only obliged to accept up to 10 pieces of 5-cent coins.

You now want to buy an item that costs **C** cents. **C** is guaranteed to be a multiple of 5 since that is a requirement in Singapore. You look into your wallet and see a certain number of each coin which you want to use to pay for the item, and it is guaranteed that you are able to pay for the item using the coins in your wallet. You also want to use as many coins as possible because you want to lighten your wallet as much as possible, because of this you also don't want to use any notes.

Taking into considering the new legal tender limit which prevents you from using more than 10 pieces of any particular type of coin, you wonder what is the maximum number of coins you can use to buy the item. You also wonder what is the number of possible ways you can use that maximum number of coins. This is difficult to do by hand so being a good programmer, you want to write a program that can do this calculation for you so you can use it any time you want to buy an item.

Good luck!

Input

The first line contains a single integer **C (5 <= N <= 1850),** representing the price of the item you want to buy in <u>cents</u>.

The next line contains 5 integers, representing the number of 5-cent, 10-cent, 20-cent, 50-cent and one-dollar coins you have in your wallet respectively. Each value will be between 0 to 100 inclusive. One dollar corresponds to 100 cents.

Output

Output two integers separated by a single space. The first integer represents the maximum number of coins you can use and the second integer represents the number of ways you can use that maximum number of coins. Both of these values should take into account that <u>you cannot use more than 10 pieces of a particular type of coin.</u> The last line of the output must contain a <u>newline character</u>.

Sample Input

```
1110
3 0 3 12 8
```

Sample Output

```
18 2
```

[1] http://www.mas.gov.sg/~/media/MAS/News%20and%20Publications/Consultation%20Papers/16%20Mar%2017%20%20Consultation%20Paper%20on%20Revised%20Legal%20Tender%20Limits%20for%20Coins.pdf

Explanation

You can use at most 18 coins for the transaction and there are 2 ways to achieve this.

1. 2 pieces of 5-cent, 10 pieces of 50-cent, 6 pieces of one-dollar
2. 3 pieces of 20-cent, 9 pieces of 50-cent, 6 pieces of one-dollar

There is a way that uses 2019 coins which is to use 2 pieces of 5-cent, 12 pieces of 50-cent and 5 pieces of one-dollar but this violates the limit of using at most 10 pieces of any particular type of coin.

Skeleton

You are given the skeleton file `CountingChange.java`. You should see a non-empty file when you open the skeleton, otherwise you might be in the wrong directory.

```java
/**
 * Name          :
 * Matric No.    :
 * PLab Acct.    :
 */

public class CountingChange {

    private void run() {
        // treat this as your "main" method
    }

    public static void main(String[] args) {
        CountingChange myCountingChange = new CountingChange ();
        myCountingChange.run();
    }
}
```

Notes

1. You should develop your program in the subdirectory **ex2** and use the skeleton file provided.
2. You must use **recursion** to solve this problem. Otherwise, you will get 0 for this problem.
3. This problem is worth 70% of the total Practical Exam marks.
4. Please be reminded that the marking scheme is:

Input	: 10%
Output	: 10%
Correctness	: 50%
Programming Style	: 30% (awarded if you score **at least 20% from the above**):

- Meaningful comments (pre- and post- conditions, comments inside the code): 10%
- Modularity (modular programming, proper modifiers [public / private]): 10%
- Proper Indentation: 5%
- Meaningful Identifiers (for both method and variable names): 5%

Compilation Error : Deduction of **50% of the total marks obtained.**