Problem A. Create the Best Pet

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Petya plays a multiplayer computer game. The hero which represents Petya in the game can have up to three pets.

Each pet has eight parameters responsible for its attack and defense power. Each parameter is an integer from 1 to 999. Numbers greater than 500 primarily increase attack power, and lesser numbers are good for defense power. Generally, the more each parameter differs from 500 to either side, the better. Furthermore, each pet has eight genetic parameters which define how it looks. Lastly, each pet has a name.

When a player decides to buy a pet in the game shop, only its name can be chosen. After that, an algorithm is run which picks power and genetic parameters randomly. Petya already bought one pet but was not satisfied: the power parameters turned out to be mediocre. He firmly decided to do whatever it takes to get a strong pet next time. But how to achieve it?

After some research, Petya got his hands on a piece of code which is responsible for pet generation, written in scripting language XyzzyLang. Here it is:

```
CreateRandomPet (Name):
    Seed := Now xor Hash (Name) xor Salt
    return Pet:
        Pet.Name := Name
        for I := 1, 2, ..., 8:
            Pet.Power[I] := GenerateRandomPower
        for J := 1, 2, ..., 8:
            Pet.Gene[J] := Random mod 5

Hash (String):
    Result := 0
    for I := 1, 2, ..., String.Length:
        Result := Result * 31 + String[I].AsciiCode
    return Result

Random:
    Seed := Seed * 1234567893 + 151515151
    return Seed

GenerateRandomPower:
    Result := 500
    for I := 1, 2, ..., 10000:
        Result := Result + Random mod 3 - 1
    return Result
```

Petya did not encounter this language before, but already learned that all numbers in the program are integers, and all calculations are performed modulo 2^{31}. As can be seen from the code, the pet's parameters are fully determined by three numbers: Now, Hash(Name), and Salt. Petya knows that the function Now returns the current moment in unix timestamp format (the number of seconds passed since January 1, 1970). The function Hash uses ASCII codes of characters (65–90 for `A`–`Z` and 97–122 for `a`–`z`). However, Petya didn't find the value of the number Salt anywhere.

According to Petya, the quality of a pet is the average deviation of its power parameters from the number 500: the sum of absolute differences $|500 - \texttt{Power[I]}|$ divided by 8. Note that this doesn't take genetic parameters into account.

Petya already came up with a good name for his next pet, and wants to create it at moment Start. However, in order to get a strong pet, he can wait from 0 to Wait seconds inclusive. Under these conditions, determine when exactly should Petya make a purchase in order to get a pet with the highest possible quality.

In all tests for this problem, the number Salt is equal to the same secret value from 0 to $2^{31} - 1$. The example shows the first pet that Petya bought.

Input

The first line contains two integers Start and Wait, followed by the word Name ($1\,535\,814\,000 \le$ Start $\le 2\,000\,000\,000$, $0 \le$ Wait ≤ 120). The word Name is from 1 to 20 characters long and can contain only lowercase and uppercase English letters.

Output

On the first line, print an integer: the moment to generate the best pet, and then a real number: the quality of this pet. The quality should be printed as precisely as possible. On the second line, print eight numbers: the power parameters of the generated pet in their respective order. On the third line, print eight numbers: the genetic parameters of the pet in their respective order.

If it is possible to generate several pets with the same maximum quality, print the one which can be generated earlier.

Example

standard input	standard output
1535814000 0 Murik	1535814000 69.250 484 467 469 363 621 504 291 503 1 3 2 3 0 0 4 2