According to Wikipedia, **autocorrelation** is the cross-correlation of a signal with itself. Having high autocorrelation values is not good for wireless signals as it increases interference. If you decide to be a wireless communication researcher in the future that deals with signals, you may face the following problem.

Consider a binary sequence $S = (s_1, \ldots, s_N)$ where each $s_i \in \{1, -1\}$. The *off-peak autocorrelations* of S are defined as:

$$C_k(S) = \sum_{i=1}^{N-k} s_i s_{i+k} \quad (k = 1 \ldots N-1)$$

and the *energy* of S is defined as:

$$E(S) = \sum_{k=1}^{N-1} C_k^2(S)$$

The problem is to assign values to the s_i such that $E(S)$ is minimized.

Just for general info, in the IEEE 802.11b standard, an 11-chip "Barker" (a researcher's name) sequence is used for the 1 and 2 Mbit/sec rates.

This sequence is S = {1, 1, 1, -1, -1, -1, 1, -1, -1, 1, -1} of length **N = 11** and it has the lowest autocorrelation values **C_k(S)** for various k in [1..10] which sum to **E(S) = 5.** This is the minimum possible **E(S)** for a binary sequence of length **N = 11**. This makes for a more uniform spectrum, and better performance in the receivers.

So that you can quickly understand this problem, here are some intermediate computations:

```
S = {1, 1, 1, -1, -1, -1, 1, -1, -1, 1, -1}
N = 11
C1 (S) =  0 ==> squared ==> 0
C2 (S) = -1                 1
C3 (S) =  0                 0
C4 (S) = -1                 1
C5 (S) =  0                 0
C6 (S) = -1                 1
C7 (S) =  0                 0
C8 (S) = -1                 1
C9 (S) =  0                 0
C10(S) = -1                 1
                           --- +
                    E(S) = 5
```

This sequence S = {1, 1, 1, -1, -1, -1, 1, -1, -1, 1, -1} can be written in Run Length Notation (RLN). Each digit in the RLN indicates the number of consecutive elements with the same sign. There are THREE consecutive +1s, THREE consecutive -1s, ONE +1, TWO consecutive -1s, ONE +1, and finally ONE -1, so the RLN is "331211".

Note that if the number of consecutive elements are between [10, 11, ..., 35], use **one character [A, B, ..., Z] instead**. We guarantee you that you will not need to deal with 36 consecutive elements or more.

TASK

For each of the following subtasks, generate a string representing the sequence in RLN. The RLN strings will be printed in 11 lines in the order of the sub-task with each line for each subtask. **Save this file as interference.out1.1**.

Subtask 0 [0 points]

Score = min(0, 10^(5 / E(S)))

This is the sample input described above (N = 11). Output the RLN that represents bit string S that minimizes interference/energy E(S) for N = 11.

Note that for this subtask (verify this by yourself): Giving RLN = "A2" in the first line of interference.out1.1 will give you score of 0.00 points for subtask 0, because the RLN sums to 12 (remember, A is 10) whereas this is a subtask for N = 11.

- RLN = "11111111111" => 1.03
- RLN = "A1" => 1.06
- RLN = "12341" => 1.24
- RLN = "111422" => 2.42
- RLN = "331211" => 10.00, the maximum possible.

Subtask 1 [10 points]

N = 20
Score = 10^(26 / E(S))

Subtask 2 [10 points]

N = 21
Score = 10^(26 / E(S))

Subtask 3 [10 points]

N = 27
Score = 10^(37 / E(S))

Subtask 4 [10 points]

N = 28
Score = 10^(50 / E(S))

Subtask 5 [10 points]

N = 40
Score = 10^(108 / E(S))

Subtask 6 [10 points]

N = 50
Score = 10^(153 / E(S))

Subtask 7 [10 points]

N = 55
Score = 10^(171 / E(S))

Subtask 8 [10 points]

N = 77
Score = 10^(366 / E(S))

Subtask 9 [10 points]

N = 80
Score = 10^(400 / E(S))

Subtask 10 [10 points]

N = 100
Score = 10^(722 / E(S))

SUBMISSION DETAILS

This is an **output-only** task, that is, all the test cases are available for you. You do not need to submit any program. Instead, submit only the necessary output. A sample **interference.out1.1** file (containing 11 lines) might look like:

```
331211
1
2
3
4
5
6
7
8
9
A
```

Of course, submitting this output will result in 0 credit being given. Before submission, you are required to gzip compress your file. To do so, use the ***tar*** command as follows:

```
tar -cvzf interference.tgz interference.out1.1
```

Submit only the compressed **.tgz** file to Mu Judge.

GRADING DETAILS

This is what the grader will do for each line of your output:

1. Convert the RLN in into bit string S (Each 0 is converted to -1).
2. Note that if the RLN is found to be invalid, i.e. the sum of individual digit values != N for the subtask, your score for that subtask is automatically 0.
3. Evaluate the bit string using the E(S) formula.
4. Evaluate your score using the marking scheme above and produce a total score for this task rounded to the nearest integer between 0 and 100.

Note that your output must **strictly follow** the output format as specified (e.g characters must be uppercase etc.). Otherwise you might risk getting a complete 0 for your entire submission even if only one line failed to follow the output format.