Codeforces Round #306 (Div. 2)

A. Two Substrings

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given string s. Your task is to determine if the given string s contains two non-overlapping substrings "AB" and "BA" (the substrings can go in any order).

Input
The only line of input contains a string s of length between 1 and 10^5 consisting of uppercase Latin letters.

Output
Print "YES" (without the quotes), if string s contains two non-overlapping substrings "AB" and "BA", and "NO" otherwise.

Sample test(s)

input
ABA
output
NO

input
BACFAB
output
YES

input
AXBYBXA
output
NO

Note
In the first sample test, despite the fact that there are substrings "AB" and "BA", their occurrences overlap, so the answer is "NO".

In the second sample test there are the following occurrences of the substrings: **BA**CF**AB**.

In the third sample test there is no substring "AB" nor substring "BA".

B. Preparing Olympiad

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have n problems. You have estimated the difficulty of the i-th one as integer c_i. Now you want to prepare a problemset for a contest, using some of the problems you've made.

A problemset for the contest must consist of at least two problems. You think that the total difficulty of the problems of the contest must be at least l and at most r. Also, you think that the difference between difficulties of the easiest and the hardest of the chosen problems must be at least x.

Find the number of ways to choose a problemset for the contest.

Input

The first line contains four integers n, l, r, x ($1 \le n \le 15, 1 \le l \le r \le 10^9, 1 \le x \le 10^6$) — the number of problems you have, the minimum and maximum value of total difficulty of the problemset and the minimum difference in difficulty between the hardest problem in the pack and the easiest one, respectively.

The second line contains n integers $c_1, c_2, ..., c_n$ ($1 \le c_i \le 10^6$) — the difficulty of each problem.

Output

Print the number of ways to choose a suitable problemset for the contest.

Sample test(s)

input
3 5 6 1 1 2 3
output
2

input
4 40 50 10 10 20 30 25
output
2

input
5 25 35 10 10 10 20 10 20
output
6

Note

In the first example two sets are suitable, one consisting of the second and third problem, another one consisting of all three problems.

In the second example, two sets of problems are suitable — the set of problems with difficulties 10 and 30 as well as the set of problems with difficulties 20 and 30.

In the third example any set consisting of one problem of difficulty 10 and one problem of difficulty 20 is suitable.

C. Divisibility by Eight

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a non-negative integer n, its decimal representation consists of at most 100 digits and doesn't contain leading zeroes.

Your task is to determine if it is possible in this case to remove some of the digits (possibly not remove any digit at all) so that the result contains at least one digit, forms a non-negative integer, doesn't have leading zeroes and is divisible by 8. After the removing, it is forbidden to rearrange the digits.

If a solution exists, you should print it.

Input

The single line of the input contains a non-negative integer n. The representation of number n doesn't contain any leading zeroes and its length doesn't exceed 100 digits.

Output

Print "NO" (without quotes), if there is no such way to remove some digits from number n.

Otherwise, print "YES" in the first line and the resulting number after removing digits from number n in the second line. The printed number must be divisible by 8.

If there are multiple possible answers, you may print any of them.

Sample test(s)

input
3454
output
YES 344

input
10
output
YES 0

input
111111
output
NO

D. Regular Bridge

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

An undirected graph is called k-*regular*, if the degrees of all its vertices are equal k. An edge of a connected graph is called a *bridge*, if after removing it the graph is being split into two connected components.

Build a connected undirected k-regular graph containing at least one bridge, or else state that such graph doesn't exist.

Input

The single line of the input contains integer k ($1 \le k \le 100$) — the required degree of the vertices of the regular graph.

Output

Print "NO" (without quotes), if such graph doesn't exist.

Otherwise, print "YES" in the first line and the description of any suitable graph in the next lines.

The description of the made graph must start with numbers n and m — the number of vertices and edges respectively.

Each of the next m lines must contain two integers, a and b ($1 \le a, b \le n$, $a \ne b$), that mean that there is an edge connecting the vertices a and b. A graph shouldn't contain multiple edges and edges that lead from a vertex to itself. A graph must be connected, the degrees of all vertices of the graph must be equal k. At least one edge of the graph must be a bridge. You can print the edges of the graph in any order. You can print the ends of each edge in any order.

The constructed graph must contain at most 10^6 vertices and 10^6 edges (it is guaranteed that if at least one graph that meets the requirements exists, then there also exists the graph with at most 10^6 vertices and at most 10^6 edges).

Sample test(s)

input
1
output
YES 2 1 1 2

Note

In the sample from the statement there is a suitable graph consisting of two vertices, connected by a single edge.

E. Brackets in Implications

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Implication is a function of two logical arguments, its value is false if and only if the value of the first argument is true and the value of the second argument is false.

Implication is written by using character '$\rightarrow$', and the arguments and the result of the implication are written as '0' ($false$) and '1' ($true$). According to the definition of the implication:

$$0 \rightarrow 0 = 1$$
$$0 \rightarrow 1 = 1$$
$$1 \rightarrow 0 = 0$$
$$1 \rightarrow 1 = 1$$

When a logical expression contains multiple implications, then when there are no brackets, it will be calculated from left to fight. For example,

$$0 \rightarrow 0 \rightarrow 0 = (0 \rightarrow 0) \rightarrow 0 = 1 \rightarrow 0 = 0.$$

When there are brackets, we first calculate the expression in brackets. For example,

$$0 \rightarrow (0 \rightarrow 0) = 0 \rightarrow 1 = 1.$$

For the given logical expression $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots \rightarrow a_n$ determine if it is possible to place there brackets so that the value of a logical expression is false. If it is possible, your task is to find such an arrangement of brackets.

Input

The first line contains integer n ($1 \le n \le 100\,000$) — the number of arguments in a logical expression.

The second line contains n numbers $a_1, a_2, ..., a_n$ ($a_i \in \{0, 1\}$), which means the values of arguments in the expression in the order they occur.

Output

Print "NO" (without the quotes), if it is impossible to place brackets in the expression so that its value was equal to 0.

Otherwise, print "YES" in the first line and the logical expression with the required arrangement of brackets in the second line.

The expression should only contain characters '0', '1', '-' (character with ASCII code 45), '>' (character with ASCII code 62), '(' and ')'. Characters '-' and '>' can occur in an expression only paired like that: ("->") and represent implication. The total number of logical arguments (i.e. digits '0' and '1') in the expression must be equal to n. The order in which the digits follow in the expression from left to right must coincide with $a_1, a_2, ..., a_n$.

The expression should be *correct*. More formally, a *correct* expression is determined as follows:

- Expressions "0", "1" (without the quotes) are correct.
- If v_1, v_2 are correct, then v_1->v_2 is a correct expression.
- If v is a correct expression, then (v) is a correct expression.

The total number of characters in the resulting expression mustn't exceed 10^6.

If there are multiple possible answers, you are allowed to print any of them.

Sample test(s)

input
4 0 1 1 0
output
YES (((0)->1)->(1->0))

input
2 1 1
output
NO

input
1 0
output

```
YES
0
```