Problem 2. Palindrome

A string $s_0 s_1 \cdots s_{m-1}$ is said to be a **palindrome** if $s_i = s_{m-1-i}$ holds for every $i \in \{0, 1, \cdots, m-1\}$.

This problem consists of two subtasks. See Submission for how to submit your code.

Subtask 1

Read n strings and test whether each of them is a palindrome.

Input format

On the first line, a nonnegative integer n.

Then $2n$ lines follow. The $(2i-1)$-th line is a positive integer representing the length of the i-th string. The $2i$-th line is the i-th string, which is guaranteed to contain **no** whitespace characters.

Output format

n lines. For every $i \in \{1, \cdots, n\}$, print `Yes` on the i-th line if the i-th string is a palindrome; print `No` otherwise.

Example

Input:

```
5
1
a
4
(())
2
[[
5
abcba
8
ab0220ba
```

Output:

```
Yes
No
Yes
Yes
Yes
```

Notes

There are no guarantees on the maximum possible value of n or the length of the strings. **Dynamic memory should be used to store each input string.**

Subtask 2

Write a program that generates a testcase for subtask 1. **You can write it using either C or Python 3.** If you use C, the standard library functions `rand` and `srand` may be helpful.

The program should accept two **command line arguments** `n` and `max_length`, representing the number of strings and the maximum length of each string, respectively. If you don't remember what **command line argument** is, go over the slides of Lecture 8.

If you use C, assume your program (the executable, not the C source code file) is named `data_gen.exe` (or `data_gen` on Linux/Mac). Your program will be run with a command like the following (with `.\` replaced by `./` on Linux/Mac):

```
.\data_gen 10 20
```

If you use Python, assume your program is named `data_gen.py`. Your program will be run with a command like the following:

```
python data_gen.py 10 20
```

In either case, your program should print a testcase with `n` strings, each of which is no longer than `max_length` characters. For example:

```
10
4
b82j
7
ll+2MlR
7
e3Erubv
3
o2d
16
u[9H($YttY$(H9[u
3
~<~
20
*]=*o|{"_s6K}GJTbA07
6
:vFEr_
6
f%w>/$
9
%59T`7z+k
```

It is guaranteed that the values of `n` and `max_length` are representable by `int`.

The generated testcase should contain both palindromes and non-palindromes. We recommend generating a palindrome with 50% probability.

Still, you should use dynamic memory instead of arrays if you use C.

Submission

Your submission should contain two files: one named `solution.c` for subtask 1, and the other named `data_gen.c` or `data_gen.py` for subtask 2. Create a zip file `submission.zip` containing them, which can be done using the following command (with `data_gen.c` replaced by `data_gen.py` if you use Python)

- For Windows:

```
Compress-Archive -Path "solution.c", "data_gen.c" -DestinationPath "submission.zip"
```

- For Mac/Linux

```
zip -r submission.zip solution.c data_gen.c
```

(You need to `cd` to the directory containing these two files first.)

Submit `submission.zip` to the OJ.

Testcases 1 ~ 15 are for subtask 1, and the rest are for subtask 2. Note that subtask 2 is run in the stage that the OJ treats as "compile time", so if you see "Compile timeout", it probably means that your program for subtask 2 is too slow.