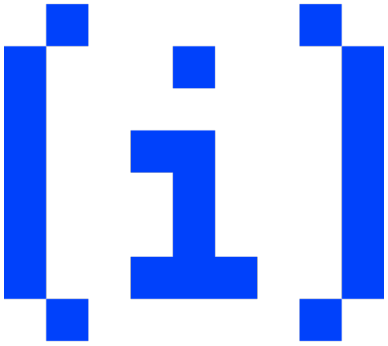


en[i]gma 0x00

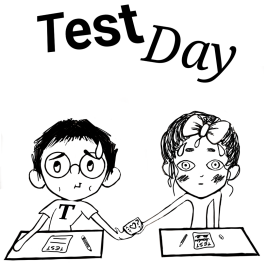
December 18, 2022

Problems

Problem	Time Limit	Memory Limit	Points
Cheating	1 sec	16MB	10
Grandma's knits	1 sec	16MB	25
Magic Squares	1 sec	16MB	25
Guess and Win!	1 sec	16MB	40
Total			100

Cheating

It is Sunday morning and Totos is stressed:



"ENIGMA begins in a few moments and I can't remember any programming language!"

Luckily his friend Annoula, who is the best at programming, is sitting next to him and intends to help him :).

The competition begins and within minutes, Annoula solves the first problem, writes it down on a piece of paper and hands it to him to copy.
She even wrote it in two of the most popular programming languages, **C** and **Python** so that he has options to choose from!
Totos immediately starts copying, but luckily, just before he hits the submit button, he realizes that something is not right.
The questions are different for each child!

He was asked the following:

"We want to write a program that reads from the input `STDIN` three positive integers N, S and T, and prints to the output `STDOUT` sets of characters `'@'` or spaces `' '` and `'\n'`.

- N ($1 \leq N \leq 10$) denotes the **number** of pieces of paper.
- S ($1 \leq S \leq 10$) denotes the **size** of the pieces of paper.
- T ($0 \leq T \leq 1$) denotes **the way of positioning** the pieces of paper, `0` for horizontal, `1` for vertical.

Attention! Each input or output line should end with the line change character `'\n'`.

Examples

1st

STDIN

```
4 3 0
```

STDOUT

```
@@@ @@@ @@@ @@@
@@@ @@@ @@@ @@@
@@@ @@@ @@@ @@@
```

2nd

STDIN

```
2 4 1
```

STDOUT

```
@@@@
@@@@
@@@@
@@@@

@@@@
@@@@
@@@@
@@@@
```

The followin is the cheat sheet ;)

C

```c
#include <stdio.h>

// read_number(), STDIN: "123 "
// |123  :   0
// 1|23  :   0 * 10 + 1 =   1
// 12|3  :   1 * 10 + 2 =  12
// 123|  :  12 * 10 + 3 = 123
// 123 | : 123
int read_number() {
    int number = 0;

    char c = getchar();
    // Xtizo ton arithmo
    while((c >= '0') && (c <= '9')) {
        int digit = c - '0';
        number = number * 10 + digit;
        c = getchar();
    }

    return number;
}

// squares_horizontal(2, 3)
// @@@ @@@
// @@@ @@@
// @@@ @@@
void squares_horizontal(int cnt, int size) {
    // Metritis grammon othonis
    for(int i = 0; i < size; i++) {
```

```
        // Metritis tegragonon
        for(int j = 0; j < cnt; j++) {

            // Metritis xaraktiron
            for(int k = 0; k < size; k++) {
                putchar('@');
            }

            putchar(' ');
        }

        putchar('\n');
    }
}

// squares_vertical(2, 3)
// @@@
// @@@
// @@@
//
// @@@
// @@@
// @@@
void squares_vertical(int cnt, int size) {
    // Metritis tegragonon
    for(int j = 0; j < cnt; j++) {

        // Metritis grammon othonis
        for(int i = 0; i < size; i++) {

            // Metritis xaraktiron
            for(int k = 0; k < size; k++) {
                putchar('@');
            }

            putchar('\n');
        }

        putchar('\n');
    }
}

// H ektelesi tou programmatos ksekina edo!
int main() {
    int cnt = read_number();
```

```
    int size = read_number();

    squares_horizontal(cnt, size);
    squares_vertical(cnt, size);

    return 0;
}
```

Python

```
from sys import stdin, stdout, exit

# read_number(), STDIN: "123 "
# |123  :   0
# 1|23  :   0 * 10 + 1 =   1
# 12|3  :   1 * 10 + 2 =  12
# 123|  :  12 * 10 + 3 = 123
# 123 | : 123
def read_number():
    number = 0

    c = stdin.read(1)
    # Xtizo ton arithmo
    while((c >= '0') and (c <= '9')):
        digit = int(c)
        number = number * 10 + digit
        c = stdin.read(1)

    return number

# squares_horizontal(2, 3)
# @@@ @@@
# @@@ @@@
# @@@ @@@
def squares_horizontal(cnt, size):
    # Metritis grammon othonis
    for i in range(0, size, 1):

        # Metritis tegragonon
        for j in range(0, cnt, 1):

            # Metritis xaraktiron
            for k in range(0, size, 1):
                stdout.write('@')

            stdout.write(' ')
```

```
        stdout.write('\n')

# squares_vertical(2, 3)
# @@@
# @@@
# @@@
#
# @@@
# @@@
# @@@
def squares_vertical(cnt, size):
    # Metritis tegragonon
    for j in range(0, cnt, 1):

        # Metritis grammon othonis
        for i in range(0, size, 1):

            # Metritis xaraktiron
            for k in range(0, size, 1):
                stdout.write('@')

            stdout.write('\n')

        stdout.write('\n')

# H ektelesi tou programmatos ksekina edo!
if __name__ == "__main__":
    cnt = read_number()
    size = read_number()

    squares_horizontal(cnt, size)
    squares_vertical(cnt, size)

    exit(0)
```

Grandma's knits

Totos loves his grandmother very much.
He always tries to spend as much time with her as possible, especially during Christmas.

Every year during this time, his grandmother prepares gifts for her children, grandchildren and great-grandchildren, which are none other than hand-knitted items!
Grandma, of course, knows that not everyone has the same tastes nor the same needs.
Some prefer scarves either short or long while others prefer a sweater in a size that fits them.

This year, Toto has decided to help his grandma in the making of the knitting patterns- surely a program that automatically determines the pattern she needs to knit is going to help her a lot!

We want to write a program that will read from the input `STDIN` a character C, followed by a positive integer N, separated by a space character `' '`, and print to the output `STDOUT` multitudes of characters `'x'` or spaces `' '` and `'\n'`.

- The C ($C \in \{$`'K'`, `'S'`, `'M'`, `'L'`$\}$) denotes the **type** of the knit.
- N ($1 \leq N \leq 4$) denotes the **number** of designs of the knitted fabric.
- When the knit type is `'K'` **scarf**, the pattern is repeated **in a row**.
- When the knit type is a **sweater** of size `'S'` or `'M'` or `'L'`, the design is repeated **horizontally**.

Attention! Each input or output line should end with the line change character `'\n'`.

Examples

1st

STDIN

```
K 4
```

STDOUT

```
x    x    x    x
 x  x      x  x
  xx        xx
 x  x      x  x
x    x    x    x
x    x    x    x
 x  x      x  x
  xx        xx
 x  x      x  x
x    x    x    x
```

```
x    x    x    x
 x  x      x  x
  xx        xx
 x  x      x  x
x    x    x    x
x    x    x    x
 x  x      x  x
  xx        xx
 x  x      x  x
x    x    x    x
```

2nd

STDIN

```
S 3
```

STDOUT

```
xx    xx    xx    xx    xx    xx
 xx  xx      xx  xx      xx  xx
  xxxx        xxxx        xxxx
   xx          xx          xx
  xxxx        xxxx        xxxx
 xx  xx      xx  xx      xx  xx
xx    xx    xx    xx    xx    xx
xx    xx    xx    xx    xx    xx
 xx  xx      xx  xx      xx  xx
  xxxx        xxxx        xxxx
   xx          xx          xx
  xxxx        xxxx        xxxx
 xx  xx      xx  xx      xx  xx
xx    xx    xx    xx    xx    xx
```

3rd

STDIN

```
M 1
```

STDOUT

```
xxxx    xxxx
 xxxx  xxxx
  xxxxxxxx
   xxxxxx
  xxxxxxxx
 xxxx  xxxx
```

```
xxxx    xxxx
xxxx    xxxx
 xxxx  xxxx
  xxxxxxxx
   xxxxxx
  xxxxxxxx
 xxxx  xxxx
xxxx    xxxx
```

4th

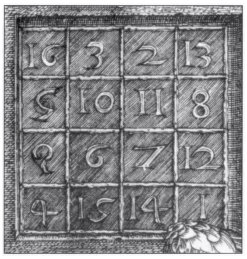
STDIN

```
L 2
```

STDOUT

```
xxxxxx    xxxxxx    xxxxxx    xxxxxx
 xxxxxx  xxxxxx      xxxxxx  xxxxxx
  xxxxxxxxxxxx        xxxxxxxxxxxx
   xxxxxxxxxx          xxxxxxxxxx
  xxxxxxxxxxxx        xxxxxxxxxxxx
 xxxxxx  xxxxxx      xxxxxx  xxxxxx
xxxxxx    xxxxxx    xxxxxx    xxxxxx
xxxxxx    xxxxxx    xxxxxx    xxxxxx
 xxxxxx  xxxxxx      xxxxxx  xxxxxx
  xxxxxxxxxxxx        xxxxxxxxxxxx
   xxxxxxxxxx          xxxxxxxxxx
  xxxxxxxxxxxx        xxxxxxxxxxxx
 xxxxxx  xxxxxx      xxxxxx  xxxxxx
xxxxxx    xxxxxx    xxxxxx    xxxxxx
```

Magic Squares



Totos needs to finally complete his term paper in computer science. The topic he got was *magic squares*. He's reading on Wikipedia:

"A magic square is an arrangement of numbers in an array of equal sets of rows and columns, where the arithmetic operation between numbers in the same row or column or diagonal of the square always returns the same result. This common result is called the magic constant of the magic square. The most common arithmetic operation in magic squares is addition between numbers, and there are other versions... "

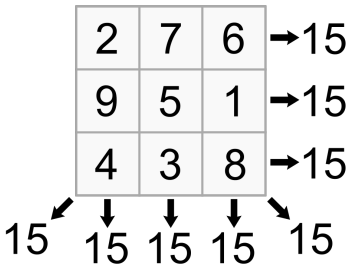


Figure 1: Example of a 3x3 magic square where the sum of the numbers in each row, column and diagonal equals 15.

The algorithm for its construction is giving him a headache. It seems easier for him to use a random number generator and select the ones that form a magic square.

We want to write a program that reads from the input `STDIN` a positive integer N, followed by N^2 **unique** positive integers P_i, separated by a space character `' '` or a line break `'\n'`, and checks whether the given square is a *magic square*.

The program will print `STDOUT` output **"NO"** if the input square does not correspond to a *magic square* or will print **"YES"** on the first line of output followed by a number S on the next line if the given square corresponds to a *magic square*.

- N ($1 \leq N \leq 100$) represents the **size of the side** of the square.
- P_i ($1 \leq P_i \leq N^2$, $1 \leq i \leq N^2$) denote **each unique number** of the square.
- S denotes the **sum** of a row, column or diagonal of the *magic square*.

Attention! Each input or output line should end with the line change character `'\n'` and the words "YES" and "NO" are written in roman characters.

Examples

1st

STDIN

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

STDOUT

```
NAI
15
```

2nd

STDIN

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

STDOUT

```
OXI
```

Guess and WIn!

The world has changed. Gambling is now the only way for people to fill their time and stay entertained.
Children are no exception and playing "Guess and Win!" is the new trend.

"Guess and Win!" is a game of chance, the aim of which is to guess a positive integer M.
Each attempt costs 1 coin unit. There is no limit to the number of attempts you can make until you get the number, but there is a limit to your financial resources :).
Kids always spend their entire allowance on the game, and now, at Christmastime, when they have their gains from the carols available, they can play up to 1000 times!

Totos, in addition to being addicted to the game, is also extremely observant. He observed that every time he loses, having predicted a number K smaller than the lucky number M, a tiny light bulb lights up with an indication of `'1'`. On the other hand, when he predicts a number K greater than M, a tiny light bulb lights up with an indication of `'2'`. In case of a victory, none of the tiny light bulbs lights up (indication `'0'`).

We want to thank Totos for the information he has provided us, by writing a script that will **always** guess M before he runs out off money. This program tries, one by one, K numbers, printing them in the output `STDOUT` in **whichever** acceptable representation, followed by a line break character `'\n'`.
After each try, it will read from `STDIN` the answer of the game, an integer L that will be a symbol for Totos' observation :).

Example

STDIN	STDOUT
	1
1	
	2
1	
	3
1	
	9
2	
	8
0	