,

Problem A. Compter les votes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Les élections approchent et tu as été chargé de compter les votes. Il y a k candidats et n votants. Ton but est de compter les votes, et de déclarer si quelqu'un a gagné les élections.

Un candidat a gagné les élections s'il a strictement plus que la moitié des votes.

Input

Sur la première ligne il y a deux entiers n et k ($1 \leq k \leq n \leq 2 \cdot 10^5$) — le nombre de votants et le nombre de candidats respectivement.

Sur la deuxième ligne il y aura n entiers a_i ($1 \leq a_i \leq k$) — a_i représente le choix du i-ème votant

Output

Tu dois imprimer deux lignes. Sur la première imprime i si le i-ème candidat a gagné les élections. Si aucun candidat n'a gagné imprime "AUCUN".

Sur la deuxième lignes, tu dois imprimer k nombres, où le i-ème représente le nombre de votes représenté par le i-ème candidat.

Examples

standard input	standard output
5 4 0 3 3 1 2	AUCUN 1 1 1 2
7 2 0 0 1 0 1 0 0	0 5 2

,

Problem B. Téléphérique

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Votre ami a une passion pour les téléphériques. Afin de pouvoir en profiter à domicile, il a acheté un modèle réduit. La vue depuis la cabine est optimale quand la pente est régulière.

C'est le cas uniquement lorsque l'inclinaison est identique à chaque point du câble. Dans le cas contraire, la cabine commencerait à se balancer, ce que Stofl désire éviter à tout prix. Il a mesuré la hauteur de chacun des piliers et a maintenant besoin de ton aide afin de construire son téléphérique.

Tous les piliers sont installés sur le sol, qui est parfaitement plat. La station de départ et la station d'arrivée peuvent être placées à une hauteur quelconque. Après avoir placé les piliers, il les arrangera afin que la pente reste uniforme.

Le téléphérique a déjà été construit par Stofl. Il y a exactement n piliers disposés avec un espacement de 1. Aide Stofl à déterminer si la vue peut être appréciée ou non.

Input

Sur la première ligne, il y a un entier n $(1 \leq n \leq 2 \cdot 10^5)$ — le nombre de téléphériques

Sur la deuxième ligne, il y a n entiers h_i $(1 \leq h_i \leq 10^9)$ — la hauteurs des piliers.

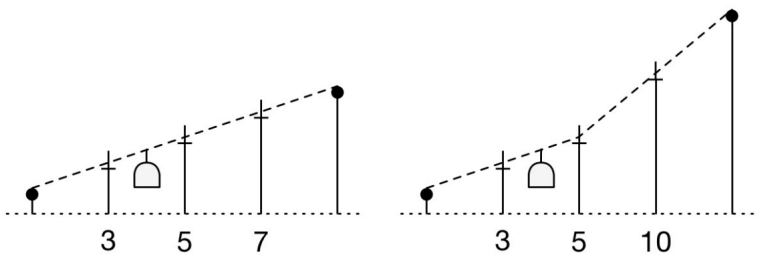
Output

Imprime "OUI"si la vue est optimale, "NON"sinon.

Examples

standard input	standard output
3 3 5 7	OUI
3 3 5 10	NON

Note



,

Problem C. Magasin

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Ton amie est propriétaire d'un magasin. Elle veut proposer autant de sortes de poisson que possible. C'est pourquoi elle achète tout ce qu'elle peut. Pour être sûr qu'elle ait la place pour tout stocker, elle garde au plus un poisson de chaque sorte.

Si un marchand rend visite à ton amie qui vend du poisson de la sorte x, elle en achète un. si elle n'en possède pas déjà une. Si un client visite le magasin et veut acheter du poisson de la sorte x, ton amie lui vend ce poisson, bien sûr – dans le cas où elle l'a en stock. Parfois, elle fait un inventaire de son magasin et compte combien de sortes de poisson elle a en stock.

Input

La première ligne de l'entrée contient n ($1 \leq n \leq 10^5$), le nombre d'interactions. Les n prochaines lignes décrivent une interaction chacune, soit avec un marchand, soit avec un client, soit pour un inventaire.

Toutes les sortes d'interaction sont identifiées par un nombre entier x ($1 \leq x \leq 10^9$).

Un nombre positif x décrit un marchand qui aimerait vendre du poisson de la sorte x.

Un nombre négatif x décrit un client qui aimerait acheter du poisson de la sorte x.

Le nombre 0 signifie que ton amie veut faire l'inventaire de son magasin.

Output

Imprime une ligne pour chaque interaction :

— "Merci !" dans le cas où elle achète le poisson d'un marchand.

— "Pas de place !" dans le cas où il n'y a pas de place pour le poisson d'un marchand.

— "Et voila !" dans le cas où elle peut vendre du poisson à un client.

— "Desole !" dans le cas où elle n'a pas le poisson demandé en stock.

— L'entier "a", le nombre de sortes de poissons actuellement en stock, dans le cas où ton amie veut faire un inventaire.

Example

standard input	standard output
9 4 2 0 4 -2 -2 -4 3 0	Merci ! Merci ! 2 Pas de place ! Et voila ! Desole ! Et voila ! Merci ! 1

,

Problem D. Extension du magasin

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Ton amie étend son magasin. Elle peut maintenant stocker un nombre arbitraire de poissons de chaque sorte. Si un marchand rend visite à ton amie qui vend du poisson de la sorte x, elle en achète un. Si un client visite le magasin et veut acheter du poisson de la sorte x, elle lui vend ce poisson, bien sûr – seulement si elle l'a en stock. Parfois, elle fait l'inventaire de son magasin et compte combien de poissons elle a en stock.

Input

La première ligne de l'entrée contient n ($1 \leq n \leq 10^5$) — le nombre d'interactions.

Les n lignes suivantes contiennent un entier x ($1 \leq x \leq 10^9$) décrivent une interaction chacune, soit avec un marchand, soit avec un client, soit pour un inventaire.

Toutes les sortes d'interaction sont identifiées par un nombre entier. Un nombre positif x décrit un marchand qui aimerait vendre du poisson de la sorte x. Un nombre négatif x décrit un client qui aimerait acheter du poisson de la sorte x. Le nombre 0 signifie que ton amie veut faire l'inventaire de son magasin.

Output

Imprime une ligne pour chaque interaction :

— "Merci !" dans le cas où elle achète le poisson d'un marchand.

— "Et voila !" dans le cas où elle peut vendre du poisson à un client.

— "Desole !" dans le cas où elle n'a pas le poisson demandé en stock.

— L'entier "a", le nombre de sortes de poissons actuellement en stock, dans le cas où ton amie veut faire un inventaire.

Example

standard input	standard output
9 4 2 0 4 -2 -2 -4 3 0	Merci ! Merci ! 2 Merci ! Et voila ! Desole ! Et voila ! Merci ! 2

,

Problème de Josèphe

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Il y a n soldats, numérotés de 1 à n sont placés en cercle. En commençant par le deuxième soldat, un soldat sur deux est éliminé (ainsi, chaque soldat qui porte un nombre pair est éliminé au premier tour).

On continue à tourner dans le cercle, toujours en éliminant un soldat sur deux, jusqu'à ce qu'il reste un soldat.

Ton but est de trouver qui est le dernier survivant.

BONUS: Analyse la complexité temporelle de ton algorithme!

Input

Il y a une ligne d'entrée contenant un entier n ($1 \leq n \leq 5 \cdot 10^5$) — le nombre initial de soldats

Output

Tu dois imprimer un entier x. Où x représente le numéro du dernier soldat en vie.

Example

standard input	standard output
6	5

Note

L'ordre d'élimination est 2, 4, 6, 3, 1, 5