Tri XOR

On vous donne un entier S et un tableau A constitué de N entiers positifs ou nuls. Les indices du tableau commencent à 1. Vous pouvez effectuer l'opération suivante sur ce tableau : choisissez un indice i ($1 \le i \le N$), choisissez l'un de ses voisins j ($1 \le j \le N$, et soit $j = i - 1$, soit $j = i + 1$), et remplacez A_i par ($A_i \oplus A_j$) où $\oplus$ est l'opération XOR bit à bit. Vous trouverez la définition du XOR à la fin de l'énoncé.

Votre objectif est de transformer A en un tableau trié :

- Si $S = 1$, alors le tableau final doit être strictement croissant : $A_i < A_{i+1}$ pour $1 \le i < N$
- Si $S = 2$, alors le tableau final doit être croissant : $A_i \le A_{i+1}$ pour $1 \le i < N$

Trouvez une séquence d'opérations qui permette d'atteindre votre objectif.
On ne vous demande pas de minimiser le nombre d'opérations, mais seulement de ne pas dépasser 40000 opérations.

Entrée

La première ligne de l'entrée contient deux entiers : N et S.
La deuxième ligne de l'entrée contient N entiers, les éléments de A.

Sortie

La première ligne de la sortie doit contenir un entier K ($0 \le K \le 40000$) - le nombre d'opérations.
Chacune des K lignes suivantes doit contenir deux entiers, décrivant une opération : le premier entier est l'indice i de l'élément qui est remplacé, et le deuxième est l'indice j de l'autre élément de l'opération. Les opérations doivent être données par ordre chronologique.

Contraintes

- $1 \le S \le 2$
- $2 \le N \le 1000$
- $0 \le A_i < 2^{20}$

Sous-tâches

1. (25 points) $2 \leq N \leq 150$, $S = 1$, Tous les éléments de A sont distincts.
2. (35 points) $2 \leq N \leq 200$, $S = 1$, Tous les éléments de A sont distincts.
3. (40 points) $2 \leq N \leq 1000$, $S = 2$

Exemples

Entrée	Sortie
5 1 3 2 8 4 1	3 1 2 4 3 5 4
5 2 4 4 2 0 1	3 3 2 4 3 5 4

Explication de la sortie du premier exemple :
[3, 2, 8, 4, 1] -> [**1**, 2, 8, 4, 1] -> [1, 2, 8, **12**, 1] -> [1, 2, 8, 12, **13**]

Explication de la sortie du deuxième exemple :
[4, 4, 2, 0, 1] -> [4, 4, **6**, 0, 1] -> [4, 4, 6, **6**, 1] -> [4, 4, 6, 6, **7**]

Lorsque vous effectuez une opération XOR entre deux bits a et b, le résultat sera 0 si a = b et 1 sinon.
Lorsque vous effectuez une opération XOR entre deux entiers a et b, on effectue un XOR pour chacun des bits d'un nombre et le bit correspondant de l'autre :
$75 \oplus 29 = 86$
$1001011 \oplus 0011101 = 1010110$

En C/C++/Java vous pouvez utiliser l'opérateur "^" pour effectuer un XOR.