Задача 2. Битоническая последовательность

Стандартный ввод
Стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт
Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Доп. ограничения	Необходимые подзадачи	Информация о проверке
1	27	$n \leq 500$		первая ошибка
2	14	$n \leq 5000$	1	первая ошибка
3	20	все числа a_i различны		первая ошибка
4	39	—	1 – 3	первая ошибка

Подзадача 1

Переберем все подходящие под условие (l, r) и для каждой такой пары сделаем проверку на битоничность.

Подзадача 2

Заметим, что для любых $l < r$ таких, что подотрезок с l по r является битоническим верно и то, что подотрезок с l по $r - 1$ — тоже битонический. Значит, мы можем зафиксировать левую границу и увеличивать правую, пока подотрезок удовлетворяет условию битоничности.

Подзадача 3

Заметим, что для битонической последовательности длины n ответ будет n^2. Давайте разобьем весь массив на битонические последовательности максимальной длины: как только наш отрезок перестаёт быть битоническим, начинаем новый в этом же месте Например, последовательности [1,2,5,3,4] нужно разбить на отрезки
(1,4) последовательность [1,2,5,3]
(4,5) последовательность [3,4]
При выводе ответа, нужно не забыть вычесть количество элементов, которые попали в два подотрезка одновременно (в этой подзадаче их количество = количество разбиений минус один).

Подзадача 4

Для полного балла нужно дополнительно обработать случай, когда подряд идет несколько одинаковых чисел. Заметим, что такие числа всегда будут находится в разных подотрезках. То есть теперь количество элементов, которые попали в два подотрезка одновременно = количество разбиений $-$(количество рядомстоящих одинаковых чисел $+1$)

```
#include <iostream>
#include <vector>
using namespace std;

typedef long long ll;

int main() {
int n;
cin >> n;

vector<ll> a(n);
vector<ll> up(n);
vector<ll> down(n);

for (int i = 0; i < n; i++) {
    cin >> a[i];
    up[i] = 1;
    down[i] = 1;
}
```

```
for (int i = 0; i < n; i++) {
    if (i > 0 && a[i - 1] < a[i]) {
    down[i] = down[i - 1] + 1; }
    }

for (int i = n − 1; i >= 0; i−−) {
    if (i + 1 < n && a[i] > a[i + 1]) {
    up[i] = up[i + 1] +1; }
}

ll cnt = 0;

for (int i = 0; i < n; i++) {
    cnt = cnt + down[i] * up[i];
}

cout ≪ cnt ≪ endl;

return 0;
}
```