**Планинско катерене**

Прайко е алпинист, но понякога се занимава и с програмиране. Като всеки един програмист, той обича да оптимизира и затова иска да автоматизира процеса за намиране на най-добра планина за катерене. За тази цел, той получава серия от числа, които представляват списък с надморски височини(коти) на различни точки. Кота, която е по–ниска от позицията пред нея и след нея, е долина. А в противоположната ситуация – кота, по-висока от съседните й , е връх. Прайко трябва да напише програма, която открива най-голямото изкачване от долина до връх, но ще има нужда от вашата помощ.

**Входни данни**

* Входните данни се състоят от един ред – цели числа, разделени със интервал.

**Изходни данни**

* Отговорът е едно цяло число – най-високото изкачване.

**Ограничения**

* Максималният брой надморски височини в списъка е 30.
* Всяка кота ще принадлежи на затворения интервал от -10 до 10.
* Всяка кота в списъка е различна от тази преди нея и тази след нея.

**Примерни тестове**

**Входни данни**

1 7 3 5 8 11 6 2

**Изходни данни**

9

**Обяснение:**

* Най-високото изкачване е от 2 до 11.

11

/ \

/ \

8 \

7 / \

/ \ / 6

/ \ 5 \

/ \ / \

/ 3 \

/ 2

1

**Входни данни**

1 2 3

**Изходни данни**

2

import java.util.Scanner;

public class Hiking {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String[] input = scanner.nextLine().split(" ");

int[] elevations = new int[input.length];

for (int i = 0; i < elevations.length; i++) {

elevations[i] = Integer.parseInt(input[i]);

}

int maxClimb = 0;

int currentClimbUp = 0;

int currentClimbDown = 0;

for (int i = 1; i < elevations.length; i++) {

if (elevations[i] > elevations[i - 1]) {

currentClimbUp += elevations[i] - elevations[i - 1];

currentClimbDown = 0;

} else {

currentClimbDown += elevations[i - 1] - elevations[i];

currentClimbUp = 0;

}

maxClimb = Math.max(maxClimb, Math.max(currentClimbUp, currentClimbDown));

}

System.out.println(maxClimb);

}

}