**Задача 3. Заек**

Лудият Шапкар току що загуби любимия си заек (Белия Заек, разбира се) някъде из редица от клетки, и сега се опитва да го намери. Клетките са номерирани с числата от 1 до . В началото заекът се намира в неизвестна клетка в редицата и всяка секунда от търсенето протича по следния начин:

1. Първо Шапкарят избира една клетка от редицата и я проверява. Наричаме тази клетка **проверената клетка.** Ако заекът е в тази клетка, търсенето приключва.
2. След това заекът избира или да остане в същата клетка, или да скочи в нейна съседна (т.е. една клетка наляво или една клетка надясно). **Забележете, че е възможно заекът да скочи в проверената клетка, ако тя е съседна; това не прекратява търсенето!**

Изборите на заекът са детерминистични спрямо настроението му. По-точно, заекът има следните две настроения:

1. ***Уплашено*** настроение – когато заекът е в това настроение, той се мести **по-далеч от проверената клетка.** Ако не е възможно да се премести по-далеч (т.е. се намира в клетка 1 или ), то той остава във същата.
2. ***Любопитно*** настроение – когато заекът е в това настроение, той се мести **по-близо до проверената клетка**. Забележете, че винаги е възможно заекът да се придвижи по-близо.

Забележете също, че заекът действа само спрямо последната проверена клетка и не взима предвид предишни проверени клетки.

Тъй като това е любимият заек на Шапкаря, той познава настроенията му много добре. По-точно, той знае, че заекът редува точно секунди в *уплашено* *настроение* и секунди в *любопитно* *настроение*. Например, ако и , настроенията на заека са дадени в редицата редицата .

Шапкарят е много притеснен за заека си и Ви моли да напишете програма rabbit.cpp, която намира списък от клетки за проверка, такива че независимо от началната позиция на заека да е гарантирано, че ще бъде намерен.

**Вход**

От първия ред на стандартния вход се въвеждат три числа: , и , описващи броя клетки и поведението на заека.

**Изход**

На първия ред от стандартния изход програмата Ви трябва да изведе – броя секунди, който отнема търсенето Ви. На втория ред програмата Ви трябва да изведе цели числа в интервала , изреждащи клетките, които се проверяват всяка секунда. Забележете, че е позволено тази редица да има повтарящи се елементи.

**Оценяване**

Нека е броя секунди във Вашето решение. Ако се опитате да проверите невалидна клетка (т.е. извън интервала ), или ако съществува случай, в който Вашият списък от проверки не намира заека, ще получите 0 точки за съответния тест както и резултат *Wrong* *Answer*. В противен случай, ако тестът дава точки, ще получите точки където:

* , ако
* , ако
* , в противен случай

Като тук:

**Ограничения**

**Информация за тестовете**

* В 8% от тестовете .
* В 12% от тестовете .
* В 8% от тестовете .

**Примерен тест**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 12 2 1 | 14  2 5 3 2 6 1 2 11 12 12 8 10 12 6 |

**Обяснение на примерния тест**

Може да се провери, че независимо от началната му позиция, дадената редица от проверки винаги намира заека. Например, да разгледаме случая, в който заека започва в клетка 8. Търсенето протича по следния начин:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Секунда*** | ***Проверена клетка*** | ***Настроение (Преди движение)*** | ***Движение на заека*** |
| 1 | 2 | Уплашено | 8 -> 9 |
| 2 | 5 | Уплашено | 9 -> 10 |
| 3 | 3 | Любопитно | 10 -> 9 |
| 4 | 2 | Уплашено | 10 -> 11 |
| 5 | 6 | Уплашено | 11 -> 12 |
| 6 | 1 | Любопитно | 12 -> 11 |
| 7 | 2 | Уплашено | 11 -> 12 |
| 8 | 11 | Уплашено | 12 -> 12 |
| 9 | 12 | Любопитно | *Намерен.* |

За това решение имаме , понеже и .

Следователно, частта от точките получена за този тест е .