Task 3: `FORENSIC`

0	1	2	3	4	5	6	7	8	9
2	5	4	4	-1	1	-1	3	0	8

The table above shows the content of an array `A`, with the indices in the top row. The array stores the *pointers* of a linear linked list. In this task, a pointer is simply an integer value. The pointer in the first node is stored in $A[0]$, that is, the value $A[0]$ indicates the location of the second node. The pointer in the second node is stored in $A[A[0]]$, whereas the pointer in the third node is stored in $A[A[A[0]]]$, and so on. If the pointer has value -1, it indicates end of the linked list. In the above example, value of $A[0]$ is 2, and thus the second pointer is stored in $A[2]$. The value of $A[2]$ is 4 and thus the third pointer is stored in $A[4]$. The value of $A[4]$ is -1, indicating that there is no more subsequent node. The sequence of pointers is shown below and the linked list has 3 nodes.

$$A[0] = 2 \quad \rightarrow \quad A[2] = 4 \quad \rightarrow \quad A[4] = -1$$

You have received the entries of an array as described above, and you are also told that one of the entries is being replaced. However, the location of that entry, and its new value are not known. Note that it is possible that the new value is the same as the original value, that is, the array remains unchanged. As a forensic expert, you want to recover the original array. To achieve that, you want to modify a single entry, so that the modified array represents a linked list with the largest number of nodes. Consider the above example:

- Changing the entry $A[4]$ to 6 leads to a linked list of 4 nodes.
- Changing the entry $A[0]$ to 7 leads to a linked list of 4 nodes.
- Changing the entry $A[0]$ to 9 leads to an invalid linked list.
- Changing the entry $A[2]$ to 7 leads to a linked list of 5 nodes.

Among all possible changes to one entry, including not changing any entry, the recovered list with 5 nodes is the largest possible. Thus, it is likely that the original entry of $A[2]$ is 7. Here, your task is to compute the size of the largest possible recovered linked list.

Input format

Your program must read from the standard input. The input consists of two lines. The only integer in the first line is N, the size of the array. The second line gives the N integers in the array, starting from $A[0]$. Each integer in the array is larger or equal to -1, and smaller than N. For the above example, the input is:

```
10
2 5 4 4 -1 1 -1 3 0 8
```

Output format

Your program must write to the standard output the number of nodes in the largest recovered linked list. For the above example, the output will be:

```
5
```

Input instances

Your program will be tested on 4 sets of input instances as follow:

1. (5 marks) All instances in this set satisfy $1 < N \leq 20$.
2. (5 marks) All instances in this set satisfy $20 < N \leq 3000$. Moreover, the number of nodes in the recovered linked list is not more than 100.
3. (5 marks) All instances in this set satisfy $20 < N \leq 3000$. Unlike the instances in the above, the size of the recovered linked list can be more than 100.
4. (10 marks) All instances in this set satisfy $3000 < N \leq 20000$.

More Samples

Be careful, there are a number of special cases. Don't miss any of them. Here are a few more samples.

Input

```
4
0 0 0 0
```

Output

```
1
```

Remark. Changing $A[0]$ to -1 give a linked list of 1 node.

Input

```
6
1 2 3 4 5 -1
```

Output

```
6
```

Remark. The input (without any change) has the largest linked list.

Input

```
10
2 5 4 4 0 1 -1 3 0 8
```

Output

```
4
```

Remark. Changing $A[4]$ to 6 gives a linked list of 4 nodes.

Input

```
10
2 5 4 4 -1 1 -1 6 0 8
```

Output

```
5
```

Remark. Changing $A[4]$ to 7 gives a linked list of 5 nodes.