Problem Statement

Your teacher has given you some problems to solve. You must first solve problem 0. After solving each problem i, you must either move on to problem i+1 or skip ahead to problem i+2. You are not allowed to skip more than one problem. For example, {0, 2, 3, 5} is a valid order, but {0, 2, 4, 7} is not because the skip from 4 to 7 is too long.

You are given a int[] **pleasantness**, where **pleasantness**[i] indicates how much you like problem i. The teacher will let you stop solving problems once the range of pleasantness you've encountered reaches a certain threshold. Specifically, you may stop once the difference between the maximum and minimum pleasantness of the problems you've solved is greater than or equal to the int **variety**. If this never happens, you must solve all the problems. Return the minimum number of problems you must solve to satisfy the teacher's requirements.

Definition

Class:	ProblemsToSolve
Method:	minNumber
Parameters:	int[], int
Returns:	int
Method signature:	int minNumber(int[] pleasantness, int variety)
(be sure your method is public)	

Constraints

- **pleasantness** will contain between 1 and 50 elements, inclusive.
- Each element of **pleasantness** will be between 0 and 1000, inclusive.
- **variety** will be between 1 and 1000, inclusive.

Examples

0)

```
{1, 2, 3}
2
Returns: 2
```

Solve the 0-th problem, and the 2-nd after it.

1)

```
{1, 2, 3, 4, 5}
4
Returns: 3
```

Obviously, the first and the last problems should be solved. Skip a problem ahead twice in a row.

2)

```
{10, 1, 12, 101}
100
Returns: 3
```

3)

```
{10, 1}
9
Returns: 2
```

4)

```
{6, 2, 6, 2, 6, 3, 3, 3, 7}
4
Returns: 2
```

You can stop after solving the first 2 problems.