**Explanation of solution**

**(rabbit)**

My solution implements multidimensional lists (matrices) which represent the field where the rabbit try to eat the carrots. Two inputs are required – size of the matrix (int) and the matrix itself (with spaces between each element). Carrots are denoted by “x”.

Example of valid input:

3

x x x

. . .

x x x

Where we have 3x3 matrix with three carrots on both 1st and 3rd rows and none on the 2nd row. Output: 2 jumps

I assume that number of jumps == number of groups of carrots (if we have two groups on the field, the rabbit has to jump 2 times, no matter the size of the field and the location of the groups of carrots).

We check every cell in our matrix. If a cell is not a carrot, we continue to the next one.

If a cell is a carrot (a “x”) we set it to a non-carrot element (e.g. a dot) and we have to determine whether it has neighbors (in order to outline the boundaries of the group). If it does not have neighbors the carrot itself is a groups so we have to increase the number of jumps by 1. If the cell has a neighboring carrot we set it to a dot as well and continue until we arrive at the last carrot of the group. Since it is the last carrot and we have “eaten” (set to dots) all its neighboring carrots, the last carrot is left without neighbors so we declare that we have found a group (jumps ++).

An interesting task is to limit the number of squares which the rabbit can overcome with one jump. 😊

I tried implementing my code in Java, but it does not work well… I am running out of time, so I will debug it later. 😊