Searching in an Array

This is a **regular task**. You must submit a PDF, which can be produced using the LaTeX template on Moodle, exported from a word processor, hand-written or any other method.

A two-dimensional array is *sorted* if each of its rows and columns is strictly increasing. For example, the following two-dimensional array is sorted.

$$\begin{bmatrix} 2 & 3 & 6 & 8 \\ 4 & 5 & 7 & 10 \\ 6 & 8 & 9 & 13 \\ 9 & 11 & 12 & 15 \end{bmatrix}$$

Given a sorted two-dimensional array A with n rows and n columns and an integer k, our task is to design an $O(n)$ algorithm that determines whether or not k appears somewhere in A.

(a) Suppose we query the top right entry $A[1][n]$ of the array. In each of the following three cases, explain which parts of the array A may contain k, and which parts of the array A definitely cannot contain k.

- **Case 1**: $A[1][n] > k$.
- **Case 2**: $A[1][n] < k$.
- **Case 3**: $A[1][n] = k$.

In each case, briefly justify your answer.

(b) Design an $O(n)$ algorithm that determines whether or not k appears in A.

Hint. *After making a query to $A[1][n]$, what are the dimensions of the search space? How could you use that information to construct the algorithm?*

(c) **(Optional)** If we instead began by examining the entry in the middle row and middle column[a], what shape does the remaining search space have? How does this make searching in a two-dimensional array more difficult than the proposed algorithm in part (b)?

[a]or one of the middle rows and columns, if the dimensions are even.

Advice.

- **Clarity:** For part (a), consider each case separately, and:
  - Provide a brief but clear, and correct description of which part(s) of the original array k can appear in. You don't have to give too strong a claim.
  - Provide one to two sentences of justification for your answer to the previous dot point.
- **(Optional) For part (c):** It is sufficient to describe the shape of the remaining search space using the proposed querying algorithm. Provide a brief but clear argument for why searching for k is more difficult than with the algorithm from part (b).

Expected length:

- For (a), up to two to three sentences per case.
- For (b), up to half a page.
- For (c), up to two to three sentences.