Find duplicate

You are given an array `A` with comparable elements. You are told that `A` contains exactly one duplicate. Thus, all elements in `A` are unique, except for *one* element, which occurs exactly twice.

(a) Assume that you are given a *sorted* array `A`, and you are told that x is the duplicate in `A`. Write an efficient procedure that prints the two positions that contains x. State the runtime complexity of the algorithm.

Input: A *sorted* array `A` with n comparable elements, and an element x which occurs exactly twice
Output: Prints the two positions $0 \leq i < j < n$ where `A[i]` $=$ `A[j]` $= x$ occur
Procedure `FindSortedIndicesOfDuplicate(A, x)`
```
| // ···
```

(b) Assume that you are given a *sorted* array `A` (and now you are not given the element that is duplicated). Write an efficient procedure that prints the two positions of the element that occurs twice. State the runtime complexity of the algorithm.

Input: A *sorted* array `A` with n comparable elements
Output: Prints the two positions $0 \leq i < j < n$ where `A[i]` $=$ `A[j]`
Procedure `FindSortedDuplicateIndices(A)`
```
| // ···
```

(c) Assume that you are given an array `A` (now you cannot assume that the array is

sorted). Write an efficient procedure that prints the two positions of the element that is duplicated. State the runtime complexity of the algorithm.

Input: An array A with comparable n elements
Output: Prints the two positions $0 \leq i < j < n$ where `A[i]` $=$ `A[j]`
Procedure `FindDuplicateIndices(A)`
 | `// ···`
