Data Structures	Completion Points Assignment 1
CCO 2103-01, Spring 2024	Due at 5:00pm, Wednesday, March 20

Hand in your solutions electronically using LearnUs.

This assignment is a completion points assignment: any reasonable attempt will receive *full* points, *even if it is incorrect.*

This is a written assignment. Your solution must be in .pdf (preferred) or .docx format. You are encouraged to use word processors, but handwritten solutions will be accepted. If you choose to handwrite your solution, you must submit a scanned version in .pdf format.

Pretend this is a closed-book exam, and try to solve these questions without referring to the lecture notes. The problem description starts on the next page.

(1) (10 points) We want to write a (static) Java method that finds the position of a given value `x` within a given array of integers that is sorted in ***descending*** order. The method declaration is as follows.

```
public static int search(int n, int[] a, int x);
```

- The parameter `x` is the value to search for.
- The parameter `a` is the array of integers.
- The parameter `n` is the number of valid entries in the given array. That is, we need to search for `x` from `a[0]`, . . . , `a[n-1]`.

If the value `x` appears within `a[0]`, . . . , `a[n-1]`, the method returns the index at which `x` appears. (If the value appears at multiple indices, the method can return any one of them.) For example, if the method returns `2103`, this means that `a[2103]` is equal to `x` and that $2103 \leq$ `n` $- 1$. If the value `x` does not appear within `a[0]`, . . . , `a[n-1]`, the method returns `-1`.

Following are four different implementations of this method, but some of these are incorrect. Determine which are correct implementations and which are not.

Implementation 1.

```
public static int search(int n, int[] a, int x) {
  if (n == 0)
    return -1;
  int left = 0;
  int right = n - 1;
  while (left < right) {
    int mid = (left + right) / 2;
    if (x > a[mid])
      right = mid - 1;
    else
      left = mid;
  }
  if (a[left] == x)
    return left;
  else
    return -1;
}
```

Implementation 2.

```
public static int search(int n, int[] a, int x) {
  int left = 0;
  int right = n;
  while (left + 1 < right) {
    int mid = (left + right) / 2;
    if (a[mid] >= x)
      left = mid;
    else
      right = mid;
  }
  if (left + 1 == right && a[left] == x)
    return left;
  else
    return -1;
}
```

Implementation 3.

```
public static int search(int n, int[] a, int x) {
  if (n == 0)
    return -1;
  int left = 0;
  int right = n - 1;
  while (right - left > 0) {
    int mid = (left + right) / 2;
    if (a[mid] == x) {
      left = mid;
      break;
    }
    if (a[mid] > x)
      left = mid + 1;
    else
      right = mid - 1;
  }
  if (a[left] == x)
    return left;
  else
    return -1;
}
```

Implementation 4.

```
public static int search(int n, int[] a, int x) {
  if (n == 0)
    return -1;
  int left = 0;
  int right = n - 1;
  while (right - left > 1) {
    int mid = (left + right + 1) / 2;
    if (a[mid] == x) {
      left = mid;
      right = mid;
      break;
    }
    if (a[mid] > x)
      left = mid + 1;
    else
      right = mid - 1;
  }
  if (a[right] == x)
    return right;
  else
    return -1;
}
```