Recursion - search

Name

1. Consider the following method, which implements a recursive binary search.

```
/** Returns an index in arr where the value x appears if x appears
*  in arr between arr[left] and arr[right], inclusive;
*  otherwise returns -1.
*  Precondition: arr is sorted in ascending order.
*                left >= 0, right < arr.length, arr.length > 0
*/
public static int bSearch(int[] arr, int left, int right, int x)
{
   if (right >= left)
   {
     int mid = (left + right) / 2;
     if (arr[mid] == x)
     {
return mid;
     }
     else if (arr[mid] > x)
     {
return bSearch(arr, left, mid - 1, x);
     }
     else
     {
return bSearch(arr, mid + 1, right, x);
     }
   }
   return -1;
}
```

The following code segment appears in a method in the same class as `bSearch`.

```
int[] nums = {0, 4, 4, 5, 6, 7};
int result = bSearch(nums, 0, nums.length - 1, 4);
```

What is the value of `result` after the code segment has been executed?

Recursion - search

A 1

B 2

C 3

D 4

E 5

Directions: Select the choice that best fits each statement. The following question(s) refer to the following information

Consider the following binarySearch method. The method correctly performs a binary search.

```
/** Precondition: data is sorted in increasing order. */
public static int binarySearch(int[] data, int target)
{
  int start = 0;
  int end = data.length - 1;
  while (start <= end)
  {
    int mid = (start + end) / 2;      /* Calculate midpoint */
    if (target < data[mid])
    {
      end = mid - 1;
    }
    else if (target > data[mid])
    {
      start = mid + 1;
    }
    else
    {
      return mid;
    }
  }
  return -1;
}
```

Recursion - search

2. Consider the following code segment.

int [ ] values = {1, 2, 3, 4, 5, 8, 8, 8};int target = 8;
What value is returned by the call binarySearch (values, target) ?

(A) -1

(B) 3

(C) 5

(D) 6

(E) 8

3.

The following question refer to the following information.
Consider the following data field and method. Method maxHelper is intended to return the largest value among the first numVals values in an array; however, maxHelper does not work as intended.

```
private int[] nums;

// precondition: 0 < numVals <= nums.length
private int maxHelper(int numVals)
{
Line 1:   int max = maxHelper(numVals - 1);

Line 2:   if (max > nums[numVals - 1])
            return max;
          else
            return nums[numVals - 1];
        }
```

Which of the following best describes the conditions under which maxHelper does not work as intended?

Recursion - search

(A) When numVals is 1

(B) When numVals is even

(C) When the elements of nums are in nonincreasing order

(D) When the elements of nums are in nondecreasing order

(E) Method maxHelper never works as intended.

Recursion - search

4. Consider the following method, which implements a recursive binary search.

```
/** Returns an index in arr where the value x appears if x appears
*  in arr between arr[left] and arr[right], inclusive;
*  otherwise returns -1.
*  Precondition: arr is sorted in ascending order.
*                left >= 0, right < arr.length, arr.length > 0
*/
public static int bSearch(int[] arr, int left, int right, int x)
{
   if (right >= left)
   {
     int mid = (left + right) / 2;
     if (arr[mid] == x)
     {
return mid;
     }
     else if (arr[mid] > x)
     {
return bSearch(arr, left, mid - 1, x);
     }
     else
     {
return bSearch(arr, mid + 1, right, x);
     }
   }
   return -1;
}
```

The following statement appears in a method in the same class as `bSearch`. Assume that `nums` is a sorted array of length 7, containing only positive integers.

```
int result = bSearch(nums, 0, nums.length - 1, -100);
```

How many times will the `bSearch` method be called as a result of executing the statement, including the initial call?

Recursion - search

(A) 1

(B) 3

(C) 4

(D) 5

(E) 7

5. Consider the following instance variable and method.

```
private int[] arr;

/** Precondition: arr contains no duplicates;
 *                the elements in arr are in ascending order.
 *  @param low an int value such that 0 ≤ low ≤ arr.length
 *  @param high an int value such that low - 1 ≤ high < arr.length
 *  @param num an int value
 */
public int mystery(int low, int high, int num)
{
  int mid = (low + high) / 2;
  if (low > high)
  {
    return low;
  }
  else if (arr[mid] < num)
  {
    return mystery(mid + 1, high, num);
  }
  else if (arr[mid] > num)
  {
    return mystery(low, mid - 1, num);
  }
  else // arr[mid] == num
  {
    return mid;
  }
}
```

What is returned by the call mystery (0, arr.length – 1, num) ?

Recursion - search

(A) The number of elements in arr that are less than num

(B) The number of elements in arr that are less than or equal to num

(C) The number of elements in arr that are equal to num

(D) The number of elements in arr that are greater than num

(E) The index of the middle element in arr