unit10-mcq

1. Consider the following method.

```
public int addFun(int n)
{
  if (n <= 0)
return 0;
  if (n == 1)
return 2;
  return addFun(n - 1) + addFun(n - 2);
}
```

What value is returned as a result of the call addFun(6) ?

(A) 10
(B) 12
(C) 16
(D) 26
(E) 32

2. 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?

(A) 1

(B) 2

(C) 3

(D) 4

(E) 5

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

```
/** Returns an index in myList where target appears,
 * if target appears in myList between the elements at indices
 * low and high, inclusive; otherwise returns -1.
 * Precondition: myList is sorted in ascending order.
 * low >= 0, high < myList.size(), myList.size() > 0
 */
public static int binarySearch(ArrayList<Integer> myList,
   int low, int high, int target)
{
     int mid = (high + low) / 2;
     if (target < myList.get(mid))
     {
         return binarySearch(myList, low, mid - 1, target);
     }
     else if (target > myList.get(mid))
     {
         return binarySearch(myList, mid + 1, high, target);
     }
     else if (myList.get(mid).equals(target))
     {
         return mid;
     }
     return -1;
}
```

Assume that `inputList` is an `ArrayList` of `Integer` objects that contains the following values.

```
[0, 10, 30, 40, 50, 70, 70, 70, 70]
```

What value will be returned by the call `binarySearch(inputList, 0, 8, 70)`?

(A) `-1`

(B) `5`

(C) `6`

(D) `7`

(E) `8`

4. Consider the following method.

```
public static int calcMethod(int num)
{
     if (num == 0)
     {
         return 10;
     }
     return num + calcMethod(num / 2);
}
```

What value is returned by the method call `calcMethod(16)` ?

(A) `10`

(B) `26`

(C) `31`

(D) `38`

(E) `41`

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;
}
```

unit10-mcq

5. 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

6. Suppose the binarySearch method is called with an array containing 2,000 elements sorted in increasing order. What is the maximum number of times that the statement indicated by / * *Calculate midpoint* * / could execute?

(A) 2,000
(B) 1,000
(C) 20
(D) 11
(E) 1

7. 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) ?

(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

8. Consider the following method.

```
// Precondition: b > 0

public int surprise(int b)

{

if ((b % 2) == 0)

{

if (b < 10)

return b;

else

return ((b % 10) + surprise(b / 10));

}

else

{

if (b < 10)

return 0;

else

return surprise(b / 10);

}

}
```

Which of the following expressions will evaluate to true ?

I. surprise(146781) == 0

II. surprise(7754) == 4

III. surprise(58216) == 16

(A) I only

(B) II only

(C) III only

(D) II and III only

(E) I, II, and III

9. Consider the following method.

```
public String recScramble(String str, int[] positions, int k)
{
if (str == null || str.length() == 0)
return "";

if (str.length() == 1)
return str;
int pos = positions[k];
String nStr = str.substring(pos, pos + 1);
str = str.substring(0, pos) + str.substring(pos + 1);
return nStr + recScramble(str, positions, k + 1);
}
```

Consider the following code segment.

```
int[] indexes = {2, 1, 1};
System.out.println(recScramble("epic", indexes, 0));
```

What is printed as a result of executing the code segment?

unit10-mcq

(A) cepi
(B) epci
(C) iecp
(D) iepc
(E) ipce

10. Consider the following method.

```
public static void showMe(int arg)
{
  if (arg < 10)
  {
    showMe(arg + 1);
  }
  else
  {
    System.out.print(arg + " ");
  }
}
```

What will be printed as a result of the call showMe(0) ?

(A) 10
(B) 11
(C) 0 1 2 3 4 5 6 7 8 9
(D) 9 8 7 6 5 4 3 2 1 0
(E) 0 1 2 3 4 5 6 7 8 9 10

unit10-mcq

11. Consider the following method.

```
/** Precondition: 0 < numVals <= nums.length */
public static int mystery(int[] nums, int v, int numVals)
{
  int k = 0;

  if (v == nums[numVals - 1])
  {
    k = 1;
  }

  if (numVals == 1)
  {
    return k;
  }
  else
  {
    return k + mystery(nums, v, numVals - 1);
  }
}
```

Which of the following best describes what the call mystery(numbers, val, numbers.length) does? You may assume that variables numbers and val have been declared and initialized.

(A) Returns 1 if the last element in numbers is equal to val; otherwise, returns 0

(B) Returns the index of the last element in numbers that is equal to val

(C) Returns the number of elements in numbers that are equal to val

(D) Returns the number of elements in numbers that are not equal to val

(E) Returns the maximum number of adjacent elements that are not equal to val

unit10-mcq

12. Consider the following method.

```
/** @param x an int value such that x >= 0
 */
public void mystery(int x)
{
  System.out.print(x % 10);
  if ((x / 10) != 0)
  {
    mystery(x / 10);
  }
  System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call mystery (1234)?

(A) 1234
(B) 4321
(C) 12344321
(D) 43211234
(E) Many digits are printed due to infinite recursion.

13. Consider the following recursive method.

```
public static void whatsItDo(String str)
{
  int len = str.length();
  if (len > 1)
  {
    String temp = str.substring(0, len - 1);
    System.out.println(temp);
    whatsItDo(temp);
  }
}
```

What is printed as a result of the call whatsItDo("WATCH") ?

unit10-mcq

(A) H

(B) WATC

(C) ATCH
ATC
AT
A

(D) WATC
WAT
WA
W

(E) WATCH
WATC
WAT
WA

14. Consider the following recursive method.

```
/** Precondition: num ≥ 0 */
public static int what(int num)
{
  if (num < 10)
  {
    return 1;
  }
  else
  {
    return 1 + what(num / 10);
  }
}
```

Assume that int val has been declared and initialized with a value that satisfies the precondition of the method. Which of the following best describes the value returned by the call what(val) ?

(A) The number of digits in the decimal representation of val is returned.

(B) The sum of the digits in the decimal representation of val is returned.

(C) Nothing is returned. A run-time error occurs because of infinite recursion.

(D) The value 1 is returned.

(E) The value val/10 is returned.

15. Consider the following recursive method.

```
public static int mystery(int n)
{
  if (n <= 1)
  {
    return 0;
  }
  else
  {
    return 1 + mystery(n / 2);
  }
}
```

Assuming that k is a nonnegative integer and $m = 2^k$, what value is returned as a result of the call mystery (m) ?

(A) 0

(B) k

(C) m

(D) $\frac{m}{2} + 1$

(E) $\frac{k}{2} + 1$

16. Consider the following recursive method.

```
public static void whatsItDo(String str)
{
  int len = str.length();
  if (len > 1)
  {
    String temp = str.substring(0, len - 1);
    whatsItDo(temp);
    System.out.println(temp);
  }
}
```

What is printed as a result of the call whatsItDo ("WATCH") ?

unit10-mcq

(A)
WATC
WAT
WA
W

(B)
WATCH
WATC
WAT
WA

(C)
W
WA
WAT
WATC

(D)
W
WA
WAT
WATC
WATCH

(E)
WATCH
WATC
WAT
WA
W
WA
WAT
WATC
WATCH

17. Consider the following recursive method.

```
public int recur(int n)
{
  if (n <= 10)
    return n * 2;
  else
    return recur(recur(n / 3));
}
```

What value is returned as a result of the call recur(27)?

(A) 8

(B) 9

(C) 12

(D) 16

(E) 18

18. Consider the following two static methods, where f2 is intended to be the iterative version of f1.

```
public static int f1(int n)
{
 if (n < 0)
 {
   return 0;
 }
 else
 {
   return (f1(n - 1) + n * 10);
 }
}
public static int f2(int n)
{
  int answer = 0;
  while (n > 0)
  {
    answer = answer + n * 10;
    n--;
  }

  return answer;
}
```

The method f2 will always produce the same results as f1 under which of the following conditions?

unit10-mcq

I. n < 0

II. n = 0

III. n > 0

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

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

Consider the following instance variable and methods. You may assume that data has been initialized with length > 0. The methods are intended to return the index of an array element equal to target, or -1 if no such element exists.

```
private int[] data;

public int seqSearchRec(int target)
{
  return seqSearchRecHelper(target, data.length - 1);
}


private int seqSearchRecHelper(int target, int last)
{
  // Line 1

  if (data[last] == target)
    return last;
  else
    return seqSearchRecHelper(target, last - 1);
}
```

19. For which of the following test cases will the call seqSearchRec(5) <u>always</u> result in an error?

I. data contains only one element.

II. data does not contain the value 5.

III. data contains the value 5 multiple times.

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I, II, and III

20. Which of the following should be used to replace // Line 1 in seqSearchRecHelper so that seqSearchRec will work as intended?

(A)
```
if (last <= 0)
  return -1;
```

(B)
```
if (last < 0)
  return -1;
```

(C)
```
if (last < data.length)
  return -1;
```

(D)
```
while (last < data.length)
```

(E)
```
while (last >= 0)
```

---

21. Consider the following method.

```
public String goAgain(String str, int index)
{
 if (index >= str.length())
   return str;

 return str + goAgain(str.substring(index), index + 1);
}
```

What is printed as a result of executing the following statement?

```
System.out.println(goAgain("today", 1));
```

(A) today
(B) todayto
(C) todayoday
(D) todayodayay
(E) todayodaydayayy

22. Consider the following `mergeSortHelper` method, which is part of an algorithm to recursively sort an array of integers.

```
/** Precondition: (arr.length == 0 or 0 <= from <= to <= arr.length)
 * arr.length == temp.length
 */
public static void mergeSortHelper(int[] arr,
   int from, int to, int[] temp)
{
     if (from < to)
     {
         int middle = (from + to) / 2;
         mergeSortHelper(arr, from, middle, temp);
         mergeSortHelper(arr, middle + 1, to, temp);
         merge(arr, from, middle, to, temp);
     }
}
```

The `merge` method is used to merge two halves of an array (`arr[from]` through `arr[middle]`, inclusive, and `arr[middle + 1]` through `arr[to]`, inclusive) when each half has already been sorted into ascending order. For example, consider the array `arr1`, which contains the values `{1, 3, 5, 7, 2, 4, 6, 8}`. The lower half of `arr1` is sorted in ascending order (elements `arr1[0]` through `arr1[3]`, or `{1, 3, 5, 7}`), as is the upper half of `arr1` (elements `arr1[4]` through `arr1[7]`, or `{2, 4, 6, 8}`). The array will contain the values `{1, 2, 3, 4, 5, 6, 7, 8}` after the method call `merge(arr1, 0, 3, 7, temp)`. The array `temp` is a temporary array declared in the calling program.

Consider the following code segment, which appears in a method in the same class as `mergeSortHelper` and `merge`.

```
int[] vals = {80, 50, 30, 20, 60, 70};
int[] temp = new int[vals.length];
mergeSortHelper(vals, 0, vals.length - 1, temp);
```

Which of the following represents the arrays merged the last time the `merge` method is executed as a result of the code segment above?

(A) `{20, 30, 50}` and `{60, 70, 80}` are merged to form `{20, 30, 50, 60, 70, 80}`.
(B) `{20, 50, 70}` and `{30, 60, 80}` are merged to form `{20, 30, 50, 60, 70, 80}`.
(C) `{20, 50, 70}` and `{30, 60, 80}` are merged to form `{20, 50, 70, 30, 60, 80}`.
(D) `{30, 50, 80}` and `{20, 60, 70}` are merged to form `{20, 30, 50, 60, 70, 80}`.
(E) `{30, 50, 80}` and `{20, 60, 70}` are merged to form `{30, 50, 80, 20, 60, 70}`.

unit10-mcq

23. 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?

(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.

24. 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 corrects the method maxHelper so that it works as intended?

(A) Insert the following statement before Line 1.
```
if (numVals == 0)
  return numVals;
```

(B) Insert the following statement before Line 1.
```
if (numVals == 1
  return nums[0];
```

(C) Insert the following statement between Line 1 and Line 2.
```
if (numVals == 0)
  return numVals;
```

(D) Insert the following statement between Line 1 and Line 2.
```
if (numVals == 1)
  return nums[0];
```

(E) Insert the following statement between Line 1 and Line 2.
```
if (numVals < 2)
  return numVals;
```

25. 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?

(A) 1

(B) 3

(C) 4

(D) 5

(E) 7

26. Consider the following method, which is intended to return the largest value in the portion of the `int` array `data` that begins at the index `start` and goes to the end of the array.

```
/** Precondition: 0 <= start < data.length */
public int maximum(int[] data, int start)
{
     if (start == data.length - 1)
     {
         return data[start];
     }
     /* missing statement */
     if (val > data[start])
     {
         return val;
     }
     else
     {
         return data[start];
     }
}
```

Which of the following can be used as a replacement for `/* missing statement */` so that the `maximum` method works as intended?

(A) `int val = maximum(data, start);`

(B) `int val = maximum(data, start - 1);`

(C) `int val = maximum(data, start + 1);`

(D) `int val = maximum(data[start - 1], start);`

(E) `int val = maximum(data[start + 1], start);`

27. Assume that methods f and g are defined as follows.

```
public int f(int x)
{
  if (x <= 0)
  {
    return 0;
  }
  else
  {
    return g(x - 1);
  }
}

public int g(int x)
{
  if (x <= 0)
  {
    return 0;
  }
  else
  {
    return (f(x - 1) + x);
  }
}
```

What value is returned as a result of the call f(6)?

(A) 0

(B) 3

(C) 6

(D) 9

(E) 12

28. Consider the following method.

```
public static int mystery(int n)
{
     if (n <= 0)
     {
         return 0;
     }
     else
     {
         return n + mystery(n - 2);
     }
}
```

What value is returned as a result of the call `mystery(9)` ?

(A) `0`

(B) `9`

(C) `16`

(D) `24`

(E) `25`

29. Consider the following data field and method.

```
private int[][] mat;

public int mystery(int r, int c)
{
  if (r != 0 || c != 0)
  {
    return (mat[r][c] + mystery(r - 1, c - 1));
  }
  else
  {
    return mat[r][c];
  }
}
```

Assume that mat is the 2-D array shown below.

	0	**1**	**2**	**3**
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

What value is returned as a result of the call mystery(2, 3)?

(A) 1

(B) 11

(C) 17

(D) 18

(E) No value is returned because mystery throws an ArrayIndexOutOfBoundsException.

unit10-mcq

30. Consider the following method.

```
// precondition: arr contains no duplicates;
//               the elements in arr are in sorted order;
//               0 ≤ low ≤ arr.length; low - 1 ≤ high < arr.length
public static int mystery(int[] arr, int low, int high, int num)
{
  int mid = (low + high) / 2;

  if (low > high)
  {
    return low;
  }
  else if (arr[mid] < num)
  {
    return mystery(arr, mid + 1, high, num);
  }
  else if (arr[mid] > num)
  {
    return mystery(arr, low, mid - 1, num);
  }
  else // arr{mid] == num
  {
    return mid;
}
```

```
}
```

How many calls to mystery (including the initial call) are made as a result of the call mystery(arr, 0, arr.length - 1, 14) if arr is the following array?

	0	1	2	3	4	5	6	7
arr	11	13	25	26	29	30	31	32

(A) 1
(B) 2
(C) 4
(D) 7
(E) 8

31. Consider the following method.

```
// precondition: x >= 0
public void mystery(int x)
{
  if ((x / 10) != 0)
  {
    mystery(x / 10);
  }

  System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call `mystery(123456)` ?

(A) 16
(B) 56
(C) 123456
(D) 654321
(E) Many digits are printed due to infinite recursion.

32. Consider the following `mergeSortHelper` method, which is part of an algorithm to recursively sort an array of integers.

```
/** Precondition: (arr.length == 0 or 0 <= from <= to <= arr.length)
 * arr.length == temp.length
 */
public static void mergeSortHelper(int[] arr,
   int from, int to, int[] temp)
{
     if (from < to)
     {
         int middle = (from + to) / 2;
         mergeSortHelper(arr, from, middle, temp);
         mergeSortHelper(arr, middle + 1, to, temp);
         merge(arr, from, middle, to, temp);
     }
}
```

The `merge` method is used to merge two halves of an array (`arr[from]` through `arr[middle]`, inclusive, and `arr[middle + 1]` through `arr[to]`, inclusive) when each half has already been sorted into ascending order. For example, consider the array `arr1`, which contains the values `{1, 3, 5, 7, 2, 4, 6, 8}`. The lower half of `arr1` is sorted in ascending order (elements `arr1[0]` through `arr1[3]`, or `{1, 3, 5, 7}`), as is the upper half of `arr1` (elements `arr1[4]` through `arr1[7]`, or `{2, 4, 6, 8}`). The array will contain the values `{1, 2, 3, 4, 5, 6, 7, 8}` after the method call `merge(arr1, 0, 3, 7, temp)`. The array `temp` is a temporary array declared in the calling program.

Consider the following code segment, which appears in a method in the same class as `mergeSortHelper` and `merge`.

```
int[] numbers = {40, 10, 20, 30};
int[] temp = new int[numbers.length];
mergeSortHelper(numbers, 0, numbers.length - 1, temp);
```

How many times will the `merge` method be called as a result of executing the code segment?

(A) 1
(B) 2
(C) 3
(D) 4
(E) 5

33. 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 = {10, 20, 30, 40, 50};
int result = bSearch(nums, 0, nums.length - 1, 40);
```

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

(A) 1

(B) 2

(C) 3

(D) 4

(E) 5

unit10-mcq

34. Consider the following method.

```
public static void mystery(int[] a, int index)
{
     if (index < a.length)
     {
         mystery(a, index + 1);
     }
     if (index > 0)
     {
         System.out.print(a[index - 1]);
     }
}
```

What is printed as a result of executing the following code segment?

```
int[] array = {6, 8, 7, 9, 5};
mystery(array, 0);
```

(A) `5978`

(B) `8795`

(C) `59786`

(D) `68795`

(E) Many digits are printed because of infinite recursion.

35. Consider the following recursive method.

```
public static void stars(int num)
{
     if (num == 1)
     {
         return;
     }
     stars(num - 1);
     for (int i = 0; i < num; i++)
     {
         System.out.print("*");
     }
     System.out.println();
}
```

What is printed as a result of the method call `stars(5)` ?

unit10-mcq

(A)
```
*****
**
***
****
*****
```

(B)
```
*
**
***
****
*****
```

(C)
```
*****
****
***
**
*
```

(D)
```
*****
****
***
**
*
```

(E)

36. The `printRightToLeft` method is intended to print the elements in the `ArrayList words` in reverse order. For example, if `words` contains `["jelly bean", "jukebox", "jewelry"]`, the method should produce the following output.

```
jewelry
jukebox
jelly bean
```

The method is shown below.

```
public static void printRightToLeft(ArrayList<String> words)
{
     if (words.size() > 0)
     {
         System.out.println(words.get(words.size() - 1));
         /* missing code */
     }
}
```

Which of the following can be used to replace `/* missing code */` so that the `printRightToLeft` method works as intended?

unit10-mcq

(A)
```
words.remove(0);
printRightToLeft(words);
```

(B)
```
words.remove(words.size());
printRightToLeft(words);
```

(C)
```
words.remove(words.size() - 1);
printRightToLeft(words);
```

(D)
```
printRightToLeft(words);
words.remove(0);
```

(E)
```
printRightToLeft(words);
words.remove(words.size() - 1);
```

37. Consider the following static method.

```
private static void recur(int n)
{
  if (n != 0)
  {
    recur(n - 2);
    System.out.print(n + " ");
  }
}
```

What numbers will be printed as a result of the call recur(7) ?

(A) -1 1 3 5 7

(B) 1 3 5 7

(C) 7 5 3 1

(D) Many numbers will be printed because of infinite recursion.

(E) No numbers will be printed because of infinite recursion.

unit10-mcq

38. Consider the following recursive method.

```
public static String recur(int val)
{
  String dig = "" + (val % 3);

  if (val / 3 > 0)
    return dig + recur(val / 3);

  return dig;
}
```

What is printed as a result of executing the following statement?

```
System.out.println(recur(32));
```

(A) 20
(B) 102
(C) 210
(D) 1020
(E) 2101

39. Consider the following recursive method.

```
private int recur(int n)
{
  if (n <= 1)
  {
    return 1;
  }
  else
  {
    return (recur(n - 2) + recur(n - 1));
  }
}
```

What value will be returned by the call recur(4)?

(A) 1
(B) 2
(C) 3
(D) 5
(E) 8

40. The `bark` method below is intended to print the string `"woof"` a total of `num` times.

```
public static void bark(int num)
{
     if (num > 0)
     {
          System.out.println("woof");
          /* missing code */
     }
}
```

Which of the following can be used to replace `/* missing code */` so that the call `bark(5)` will cause `"woof"` to be printed five times?

unit10-mcq

(A) `bark(num - 5);`

(B) `bark(num - 1);`

(C) `bark(num);`

(D) `bark(num + 1);`

(E) `bark(num + 5);`

41. Consider the following method.

```
/* Precondition: j <= k */
public static void mystery(int j, int k)
{
     System.out.println(j);
     if (j < k)
     {
         mystery(j + 1, k);
     }
}
```

Which of the following best describes the behavior of the `mystery` method?

(A) It repeatedly prints the value `j` due to infinite recursion.

(B) It prints the initial value of `j` a total of `k` times.

(C) It prints the initial value of `k` a total of `j` times.

(D) It prints the integers from `j` to `k`, inclusive, in order from least to greatest.

(E) It prints the integers from `j` to `k`, inclusive, in order from greatest to least.

42. Consider the following two methods, which are intended to return the same values when they are called with the same positive integer parameter `n`.

```
public static int mystery1(int n)
{
     if (n > 1)
     {
         return 5 + mystery1(n - 1);
     }
     else
     {
         return 1;
     }
}

public static int mystery2(int n)
{
     int total = 0;
     int x = 1;
     while (x < n)
     {
         total += 5;
         x++;
     }
     return total;
}
```

Which, if any, of the following changes to `mystery2` is required so that the two methods work as intended?

(A) The variable `total` should be initialized to `1`.

(B) The variable `x` should be initialized to `0`.

(C) The condition in the `while` loop header should be `x < n - 1`.

(D) The condition in the `while` loop header should be `x <= n`.

(E) No change is required; the methods, as currently written, return the same values when they are called with the same positive integer parameter `n`.

43. Consider the following recursive method.

```
/** Precondition: 0 <= numVals <= nums.length */
public static int mystery(int[] nums, int v, int numVals)
{
     if (numVals == 0)
     {
         return 0;
     }
     else if (v == nums[numVals - 1])
     {
         return 1 + mystery(nums, v, numVals - 1);
     }
     else
     {
         return mystery(nums, v, numVals - 1);
     }
}
```

Which of the following best describes the value returned by the call `mystery(nums, v, nums.length)` ?

(A) The value `0` is returned.

(B) The value `1` is returned.

(C) The number of times that `v` occurs in `nums` is returned.

(D) The number of times that `numVals` occurs in `nums` is returned.

(E) Nothing is returned. A runtime error occurs because of infinite recursion.

44. Consider the following method.

```
public static void strChange(String str)
{
     if (str.length() > 0)
     {
         strChange(str.substring(1));
         System.out.print(str.substring(0, 1));
     }
}
```

Which of the following best describes the behavior of the method?

(A) It prints the first character of `str`.

(B) It prints the last character of `str`.

(C) It prints the characters of `str` in the order they appear.

(D) It prints the characters of `str` in reverse order.

(E) It prints nothing due to infinite recursion.

unit10-mcq

45. Consider the following recursive method.

```
public static boolean recurMethod(String str)
{
     if (str.length() <= 1)
     {
         return true;
     }
     else if (str.substring(0, 1).compareTo(str.substring(1, 2)) > 0)
     {
         return recurMethod(str.substring(1));
     }
     else
     {
         return false;
     }
}
```

Which of the following method calls will return `true` ?

(A) `recurMethod("abcba")`

(B) `recurMethod("abcde")`

(C) `recurMethod("bcdab")`

(D) `recurMethod("edcba")`

(E) `recurMethod("edcde")`

46. Consider the following method.

```
public static int mystery(ArrayList<Integer> numList)
{
     if (numList.size() == 0)
     {
         return 0;
     }
     else
     {
         int val = numList.remove(0);
         return val + mystery(numList);
     }
}
```

Which of the following best describes the value returned by the method?

unit10-mcq

(A) It returns the value of the first element in `numList`.

(B) It returns the value of the last element in `numList`.

(C) It returns the sum of the elements in `numList`.

(D) It returns `0` regardless of the contents of `numList`.

(E) It returns nothing due to infinite recursion.