English name: ____________________

Unit 10 MCQ: Recursion

This quiz has 12 questions.

1. Consider the following recursive method, which is intended to display the binary equivalent of a decimal number. For example, `toBinary(100)` should display `1100100`.

```
public static void toBinary(int num) {
   if(num < 2) {
      System.out.print(num);
   } else {
      /* missing code */
   }
}
```

Which of the following can replace *`/* missing code */`* so that `toBinary` works as intended?

Ⓐ
```
System.out.print(num % 2);
toBinary(num / 2);
```

Ⓑ
```
System.out.print(num / 2);
toBinary(num % 2);
```

Ⓒ
```
toBinary(num % 2);
System.out.print(num / 2);
```

Ⓓ
```
toBinary(num / 2);
System.out.print(num % 2);
```

Ⓔ
```
toBinary(num / 2);
System.out.print(num / 2);
```

2. Consider the following recursive method, which is intended to return a String with any consecutive duplicate characters removed. For example, `removeDupChars("aabcccd")` returns `"abcd"`.

```
public static String removeDup(String str) {
   if(str == null || str.length() <= 1) {
      return str;
   } else if(str.substring(0,1).equals(
             str.substring(1,2))) {
      return removeDup(str.substring(1));
   } else {
      /* missing code */
   }
}
```

Which of the following can replace *`/* missing code */`* so that `removeDup` works as intended?

Ⓐ
```
return removeDup(str.substring(2));
```

Ⓑ
```
return removeDup(str.substring(1)) +
       str.substring(0,1);
```

Ⓒ
```
return removeDup(str.substring(2)) +
       str.substring(1,2);
```

Ⓓ
```
return str.substring(0,1) +
       removeDup(str.substring(1));
```

Ⓔ
```
return str.substring(1,2) +
       removeDup(str.substring(2));
```

3. Consider the following method, which is intended to return the sum of all the even digits in its parameter `num`. For example, `sumEvens(15555234)` should return `6`, the sum of `2` and `4`.

```
/** Precondition: num >= 0 */
public static int sumEvens(int num) {
   if(num < 10 && num % 2 == 0) {
      return num;
   } else if (num < 10) {
      return 0;
   } else if (num >= 10 && num % 2 == 0) {
      /* missing statement */
   } else {
      return sumEvens(num / 10);
   }
}
```

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

Ⓐ
```
return sumEvens(num % 10);
```

Ⓑ
```
return sumEvens(num / 10);
```

Ⓒ
```
return num % 10 + sumEvens(num % 10);
```

Ⓓ
```
return num % 10 + sumEvens(num / 10);
```

Ⓔ
```
return num / 10 + sumEvens(num % 10);
```

4. Consider the following method.

```
/** Precondition: n > 0 */
public static void mystery(int n) {
   System.out.print(n + " ");
   if(n > 1) {
      mystery(n – 1);
   }
}
```

Which of the following best describes the output produced by the method call `mystery(val)`?

Ⓐ All integers from `1` to `val`, separated by spaces

Ⓑ All integers from `val` to `1`, separated by spaces

Ⓒ The digits of `val` in their original order, separated by spaces

Ⓓ The digits of `val` in reverse order, separated by spaces

Ⓔ The digits of `val`, then a space, then the first digit of `val`

5. Consider the following recursive method

```
public static String doSth(String str) {
   if(str.length() < 1) {
      return "";
   } else {
      return str.substring(0,1) +
             doSth(str.substring(1));
   }
}
```

Which of the following best describes the result of the call `doSth(myString)`?

Ⓐ The method call returns a `String` containing the contents of `myString` unchanged.

Ⓑ The method call returns a `String` containing the contents of `myString` with the order of the characters reversed from their order in `myString`.

Ⓒ The method call returns a `String` containing all but the first character of `myString`.

Ⓓ The method call returns a `String` containing only the first and second characters of `myString`.

Ⓔ The method call returns a `String` containing only the first and last characters of `myString`.

6. Consider the following recursive method.

```
/** Precondition: n > 0 */
public static int calc(int n) {
   if(n <= 9) {
      return n;
   } else {
      return calc(n / 10);
   }
}
```

Which of the following best describes the value returned by the method call `calc(num)`?

Ⓐ The `int` value `9`

Ⓑ The leftmost digit of `num`

Ⓒ The rightmost digit of `num`

Ⓓ The number of digits in `num`

Ⓔ The result of the integer division of `num` by `10`

Unit 10 MCQ: Recursion

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

```
/** Precondition:
 * arr.length == temp.length
 * (arr.length == 0 or
 *  0 <= from <= t0 <= arr.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 segment, which appears in a method in the same class as `mergeSortHelper` and merge.

```
int[] arr1 = {9, 1, 3, 5, 4);
int[] temp = new int[arr1.length];
mergeSortHelper(arr1,0,arr1.length-1,temp);
```

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

Ⓐ `{9}` and `{1}` are merged to form `{1, 9}`

Ⓑ `{1, 9}` and `{3}` are merged to form `{1, 3, 9}`

Ⓒ `{1, 9}` and `{5, 4}` are merged to form `{1, 4, 5, 9}`

Ⓓ `{1, 3, 9}` and `{5}` are merged to form `{1, 3, 5, 9}`

Ⓔ `{1, 3, 9}` and `{4, 5}` are merged to form `{1, 3, 4, 5, 9}`

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

```
/** Returns an index in arr where val
 *  appears, if val appears in arr
 *  between arr[low] and arr[high],
 *  inclusive; otherwise returns -1.
 *  Preconditions:
 *    arr is sorted in ascending order,
 *    low >= 0, high < arr.length,
 *    arr.length > 0
 */
public static int bSearch(int[] arr,
         int low, int high, int val) {
   if(low > high) {
      return -1;
   }
   int middle = (low + high) / 2;
   if(val == arr[middle]) {
      return middle;
   } else if(val < arr[middle]) {
      return bSearch(arr,low,middle-1,val);
   } else {
      return bSearch(arr,middle+1,high,val);
   }
}
```

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

```
int[] arr = {2, 3, 12, 34, 54};
int result =
  binaryS(arr,0,arr.length-1,5);
```

If the first call to `bSearch` is the call in the code segment above, with `low = 0` and high = 4, which, if any, of the following shows the values of `low` and `high` when `bSearch` is called for the third time?

Ⓐ `low = 0, high = 1`

Ⓑ `low = 0, high = 2`

Ⓒ `low = 1, high = 1`

Ⓓ `low = 2, high = 1`

Ⓔ The method returns to the calling code segment before the third call to `bSearch`.

Unit 10 MCQ: Recursion

9. 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.
 *  Preconditions:
 *    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 target = 10;
int[] arrWithDups =
  {2, 3, 7, 8, 10, 10, 10, 20};
int arrIndex = bSearch(
          arrWithDups,
          0,arrWithDups.length-1,
          target);
```

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

Ⓐ 4

Ⓑ 5

Ⓒ 6

Ⓓ 7

Ⓔ 10

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

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

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

```
ArrayList<Integer> theList =
                 new ArrayList<Integer>();
for(int k = 10; k < 65; k = k + 5) {
    theList.add(k);
}
int result = bSearch(theList,
                     0,theList.size()-1,45);
```

Including the call to `bSearch` in the last statement of the given code segment, how many times will `bSearch` be called before a value is returned?

Ⓐ 1

Ⓑ 2

Ⓒ 3

Ⓓ 4

Ⓔ 8

Unit 10 MCQ: Recursion

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

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

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

```
String[] words = { "arc", "bat", "cat",
   "dog", "egg", "fit", "gap", "hat"};
int index = bSearch(words,
   0, words.length-1, "hat");
```

How many times will `"right"` be printed when the code segment is executed?

Ⓐ 1

Ⓑ 2

Ⓒ 3

Ⓓ 7

Ⓔ 8

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

```
/** Returns an index in nums where val
 *  appears, if val appears in nums
 *  between nums[lo] and nums[hi],
 *  inclusive; otherwise returns -1.
 *  Preconditions:
 *    theList is sorted in ascending order;
 *    lo >=0, hi < nums.length,
 *    nums.length > 0
 */
public static int bSearch(int[] nums,
      int lo, int hig, int val)
{
  if(hi > lo) {
    int mid = (lo + hi) / 2;
    if(nums[mid] == val) {
       return midd;
    }
    if(nums[mid] > target) {
       return bSearch(nums, lo, mid-1, val);
    } else {
       return bSearch(nums, mid+1, hi, val);
    }
  }
  return -1;
}
```

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

```
int target = 3;
int[] nums =
      {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
int tIndex =
      bSearch(nums,0,nums.length-1,target);
```

Including the call to `bSearch` in the last statement of the given code segment, how many times will `bSearch` be called as a result of executing the code segment above?

Ⓐ 1

Ⓑ 2

Ⓒ 3

Ⓓ 4

Ⓔ 5