Quiz 11

Due Nov 18 at 12pm **Points** 10 **Questions** 10 **Time Limit** None

Instructions

There is no time limit, but you may only make one submission.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	53 minutes	9 out of 10

Score for this quiz: **9** out of 10
Submitted Nov 15 at 3:39pm
This attempt took 53 minutes.

Question 1

1 / 1 pts

Consider the helper method `reversePrint`, which uses recursion to display in reverse the elements in a section of an array limited by the `firstIndex` and `lastIndex` arguments. What statement should be used to complete the recursive method?

```
public static void reversePrint(int[] array, int firstIndex, int lastIndex)
{
    if (firstIndex < lastIndex)
    {
        reversePrint(array, firstIndex + 1, lastIndex);
    }

    System.out.println(_________________);
}

public static void main(String[] args)
{
    int [] numbers = { 4, 7, 1, 0, 2, 7 };
    reversePrint(numbers, 0, numbers.length - 1);
}
```

- `array[lastIndex - 1]`

- `array[lastIndex]`
- `array[firstIndex + 1]`

Correct!

- (•) `array[firstIndex]`

Question 2

1 / 1 pts

What is required to make a recursive method successful?

I. special cases that handle the simplest computations directly

II. a recursive call to simplify the computation

III. a mutual recursion

Correct!

- (•) I and II only
- ( ) I, II, and III
- ( ) I only
- ( ) II only

Question 3

1 / 1 pts

Consider the `square()` method shown below that takes a non-negative `int` argument. Complete the code for the `square()` method so that it correctly calls the `squareHelper()` method to produce the square of `n`.

```
public int square(int n)
{
    ____________________;
}

public int squareHelper(int c, int n)
{
    if (c == 1)
    {
        return n;
```

```
    }
    else
    {
        return n + squareHelper(c - 1, n);
    }
}
```

- `return square(n)`
- `return squareHelper(n, n - 1)`
- `return squareHelper(n - 1, n)`

Correct!

- `return squareHelper(n, n)`

Question 4

1 / 1 pts

Consider the `getArea()` method from the textbook shown below.

```
public int getArea()
{
    if (width <= 0)                                          // line #1
    {
        return 0;                                            // line #2
    }
    else if (width == 1)                                     // line #3
    {
        return 1;                                            // line #4
    }
    else
    {
        Triangle smallerTriangle = new Triangle(width - 1); // line #5
        int smallerArea = smallerTriangle.getArea();         // line #6
        return smallerArea + width;                          // line #7
    }
}
```

Where is/are the recursive call(s)?

- line #1

Correct!

- line #6
- lines #1 and #3

line #3

Question 5

1 / 1 pts

Consider the `permutations()` method from the textbook, which is intended to return all permutations of the `word` passed in as a parameter. How does the `permutations()` method simplify its input for the recursive call?

```
public static ArrayList<String> permutations(String word)
{
    ArrayList<String> result = new ArrayList<String>();
    if (word.length() == 0)
    {
        result.add(word);
        return result;
    }
    else
    {
        for (int i = 0; i < word.length(); i++) // line #4
        {
            String shorter = word.substring(0, i) + word(substring(i + 1));
            ArrayList<String> shorterPermutations = permutations(shorter);
            for (String s : shorterPermutations)
            {
                result.add(word.charAt(i) + s);
            }
        }
        return result;
    }
}
```

- It finds permutations of a shorter word by removing both the first and last character.
- It finds permutations of a shorter word by removing the first character.

Correct!

- It finds permutations of shorter words formed by removing the *i*th character.
- It finds permutations of a shorter word by removing the last character.

Question 6

0 / 1 pts

What is the most likely input to a Merge operation in a Merge-Sort algorithm?

You Answered

- [x] An array

Correct Answer

- [ ] Two sorted arrays
- [ ] Two arrays
- [ ] A sorted array

Question 7

1 / 1 pts

Which of the following algorithms most naturally involve recursion?

I. Binary Search

II. Insertion Sort

III. Merge Sort

Correct!

- [x] I and III only
- [ ] II only
- [ ] I only
- [ ] II and III only

Question 8

1 / 1 pts

What is the best way to describe the complexity of the Insertion Sort algorithm?

- [ ] Log-Linear
- [ ] Constant time

Linear

Correct!

Quadratic

Question 9

1 / 1 pts

What is the complexity of the Binary Search algorithm?

O(n^2) [Note: n^2 means n-squared]

Correct!

O(log n)

O(n log n)

O(n)

Question 10

1 / 1 pts

Which of the following is most true about algorithm complexity, assuming large tasks?

Correct!

Polynomial-time algorithms are generally practical while exponential-time algorithms are typically not practical

Linear-time algorithms are faster than both exponential-time and logarithmic-time algorithms

An algorithm of complexity O(n^5) is considered more complex (taking more time) than an algorithm of complexity O(5^n)

Exponential-time algorithms are generally practical while polynomial-time algorithms are typically not practical

Quiz Score: **9** out of 10