Answer Key — Week 7 Parsons Problems

For TAs only — do not distribute

Problem 1 — Reverse a string

```java
public class ReverseString {
    static String reverse(String s) {
        if (s.isEmpty()) return s;
        return reverse(s.substring(1)) + s.charAt(0);
    }

    public static void main(String[] args) {
        String word = args[0];
        System.out.println(reverse(word));
    }
}
```

DISTRACTOR `return reverse(s) + s.charAt(0);`

Recurses on `s` itself instead of `s.substring(1)` — the input never shrinks, so the recursion never reaches the base case and the program crashes with a stack overflow.

DISTRACTOR `return s.charAt(0) + reverse(s.substring(1));`

Puts the first character **before** the recursive call instead of after — this rebuilds the string in its original order, so the program prints the word unchanged instead of reversed.

Problem 2 — Check for a palindrome

```java
public class IsPalindrome {
    static boolean isPalindrome(String s) {
        if (s.length() <= 1) return true;
        if (s.charAt(0) != s.charAt(s.length() - 1)) return false;
        return isPalindrome(s.substring(1, s.length() - 1));
    }

    public static void main(String[] args) {
        String word = args[0];
        System.out.println(isPalindrome(word));
    }
}
```

DISTRACTOR `if (s.length() == 0) return true;`

Only handles the empty string as a base case, missing the single-character case — once the recursion narrows down to one character, `s.substring(1, 0)` throws a `StringIndexOutOfBoundsException`.

DISTRACTOR `return isPalindrome(s.substring(0, s.length() - 1));`

Only drops the last character and keeps the first — the recursion never removes the character already checked, so it reports some genuine palindromes (e.g. `"abcba"`) as not palindromes.