



## Recursion & Java Strings

Problem Set: Assignment #1 due Thursday, February 9

B-1



## Recursion

- One of the important problem solving strategies
- To solve a problem recursively:
  - If it is a simple ("base") case, then give the answer immediately
  - Otherwise express the solution as a smaller similar problem ("assume you know the solution to the smaller version, then compose the solution to the larger problem")
- A good recursive solution often reflects deep insight into a problem

B-2



## The Canonical Example: Factorial

$n! = n * (n-1) * (n-2) * \dots * 1$  for any integer  $n > 0$   
 $0! = 1$

- A recursive definition of factorial(n)

$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * \text{factorial}(n-1) & \text{if } n > 0 \end{cases}$$

factorial(n) =

factorial(n-1)

B-3



## The String Class

- Strings are objects in Java

```
String s1;  
s1 = "lovelace";  
String s2 = new String("lovelace");  
String s3 = "";
```

- equals() instance method

```
s1.equals(s2)  
s1.equals(s3)  
s1.equals("lovelace")  
"lovelace".equals(s1)
```

B-4



## More String Methods

---

```
String s1 = "lovelace";
s1.length()
s1.charAt(2)
s1.charAt(8)
s1.indexOf('e')
s1.indexOf('e', 2)
s1.indexOf('l', 2)
s1.indexOf('v', 5)
s1.substring(2, 6)
s1.substring(4)
s1.substring(2, 3)
"I " + s1.substring(0, 4) + " cs230" + '.'
s1.replace('l', 'c')
```

B-5



## Reversing a String

---

```
reverse("madam")
reverse("sir")
reverse("abcd")
/** returns a reversed version of the input string */
public static String reverse(String s) {

}
```

B-6



## Reversing a String

---

```
reverse("madam")
reverse("sir")
reverse("abcd")
/** returns a reversed version of the input string */
public static String reverse(String s) {

    if (s.length() < 2) return s; // Base case

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

}
```

B-7



## Check if Two Strings are Equal

---

```
equals("madam", "madam")
equals("sir", "madam")

public static boolean equals(String s1, String s2) {

}

}
```

B-8



## Check if Two Strings are Equal

```

equals("madam", "madam")
equals("sir", "madam")

public static boolean equals(String s1, String s2) {

    if ((s1.length() == 0) && (s2.length() == 0))
        return true;

    else if (s1.length() == 0 || (s2.length() == 0))
        return false;

    else if (s1.charAt(0) != s2.charAt(0))
        return false;

    else return equals(s1.substring(1), s2.substring(1));
}

```

B-9



## Replacing a Character

```

replaceChar("madam", 'm', 'r')
replaceChar("sir", 'x', 'y')
replaceChar("abcd", 'c', 'a')
/** returns a String in which all occurrences of the input character
 * "oldC" in the input String S have been replaced with "newC"
 */
public static String replaceChar(String s, char oldChar, char newChar) {

}

```

B-10



## Replacing a Character

```

replaceChar("madam", 'm', 'r')
replaceChar("sir", 'x', 'y')
replaceChar("abcd", 'c', 'a')
/** returns a String in which all occurrences of the input character
 * "oldC" in the input String S have been replaced with "newC"
 */
public static String replaceChar(String s, char oldChar, char newChar) {
    // Base case. s is the empty string
    if (s.length() == 0) return s;

    else if (s.charAt(0) == oldChar) // First char of s needs replacing
        return (newChar +
                replaceChar(s.substring(1), oldChar, newChar));

    else // First char of s does not need replacing
        return (s.charAt(0) +
                replaceChar(s.substring(1), oldChar, newChar));
}

```

B-11



## Checking for Duplicate Characters

```

isDuplicate("madam")
isDuplicate("sir")
isDuplicate("lovelace")
/** returns true if the input String S contains at least one
 * character that is duplicated
 */
public static boolean isDuplicate(String s) {

}

```

B-12



## Checking for Duplicate Characters

```

isDuplicate("madam")
isDuplicate("sir")
isDuplicate("lovelace")
/** returns true if the input String S contains at least one
 * character that is duplicated
 */
public static boolean isDuplicate(String s) {
    if (s.length() < 2) return false; // Base case

    else {
        String s1 = s.substring(1);
        if (s1.indexOf(s.charAt(0)) > -1)
            return true;
        else
            return isDuplicate(s1);
    }
}

```

B-13



## Alphabetizing

```

insertCharIntoAlphabetizedString("abccdddfgg", 'e')
insertCharIntoAlphabetizedString("lot", 'b')
insertCharIntoAlphabetizedString("go", 'o')
/** Returns a string consisting of the characters in s and c,
 * all in alphabetical order.
 * The input string s must be in alphabetical order.
 */
public static String insertCharIntoAlphabetizedString(String s, char c) {
}
alphabetize("lovelace")
alphabetize("q")
alphabetize("structures")
/** Returns a string consisting of the
 * characters in s in alphabetical order. */
public static String alphabetize(String s) {
}

```

B-14



## insertCharIntoAlphabetizedString

```

insertCharIntoAlphabetizedString("abccdddfgg", 'e')
insertCharIntoAlphabetizedString("lot", 'b')
insertCharIntoAlphabetizedString("go", 'o')
/** Returns a string consisting of the characters in s and c,
 * all in alphabetical order.
 * The input string s must be in alphabetical order.
 */
public static String insertCharIntoAlphabetizedString(String s, char c) {
    if (s.length() == 0) return s + c; // Base case

    else {
        if (c < s.charAt(0)) return (c + s);
        else return (s.charAt(0) +
            insertCharIntoAlphabetizedString(s.substring(1), c));
    }
}

```

B-15



## Alphabetizing

```

alphabetize("lovelace")
alphabetize("q")
alphabetize("structures")
/** Returns a string consisting of the
 * characters in s in alphabetical order.
 */
public static String alphabetize(String s) {
    if (s.length() < 2) return s; // Base case

    else return insertCharIntoAlphabetizedString(
        alphabetize(s.substring(1)), s.charAt(0));
}

```

B-16