Eric Roberts and Jerry Cain
CS 106J
Handout #25
May 3, 2017

Section Handout #4: String Processing

Portions of this handout by Eric Roberts, Patrick Young, Jeremy Keeshin and Nick Troccoli

1. Adding commas to numeric strings

When large numbers are written out on paper, it is traditional—at least in the United States—to use commas to separate the digits into groups of three. For example, the number one million is usually written in the following form:

```
1,000,000
```

To make it easier for programmers to display numbers in this fashion, implement a function

```
function addCommasToNumericString(digits)
```

that takes a string of decimal digits representing a number and returns the string formed by inserting commas at every third position, starting on the right. For example, the code below should produce the following outputs.

```
addCommasToNumericString("17")          returns 17
addCommasToNumericString("1001")        returns 1,001
addCommasToNumericString("12345678")    returns 12,345,678
addCommasToNumericString("999999999")   returns 999,999,999
```

2. Deleting characters from a string

Write a function

```
function removeAllOccurrences(str, ch)
```

that removes all occurrences of the character **ch** from the string **str**. For example, your function should return the values shown:

```
removeAllOccurrences("This is a test", 't'))  returns "This is a es"
removeAllOccurrences("Summer is here!", 'e')  returns "Summr is hr!"
removeAllOccurrences("---0---", '-')          returns "0"
```

Toolbox (more documentation can be found on page 200 of the course reader):

```
str.length
str.charAt(i)
str.substring(start)
str.substring(start, end)
str.indexOf(pattern)
str.toLowerCase()
str.toUpperCase()
str.startsWith(prefix)
str1 + str2
str1 === str2
```

3. Converting a string to alternating capital letters

Write a function

```
function altCaps(str)
```

which converts a string to alternating capital letters, meaning you alternate between uppercase and lowercase. This style of typing was prevalent on the internet in the late 90s. For example:

```
altCaps("hello")                returns "hElLo"
altCaps("section is awesome") returns "sEcTiOn Is AwEsOmE"
```

Note that characters that are not letters are not changed and do not affect the alternating sequence of uppercase and lowercase letters. You can assume you have a function `isLetter(ch)` that returns `true` if `ch` is a boolean and `false` otherwise. Try to write this helper function if you have extra time!

4. An ineffective cipher

One of the most commonly known ciphers is called the shift cipher, or the Caesar cipher. The Caesar cipher tells us to shift the alphabet some number of times to the right (wrapping around at Z) to form the plaintext and the ciphertext [1]. For example, a 3-shift looks like:

```
plaintext:  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ciphertext: X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
```

We can then encrypt our message by replacing every `A` in our message with an `X`, and so on. However, as eager but budding cryptologists we haven't learned of the Caesar cipher yet, and *we misinterpreted what a shift cipher was*.

Our ineffective cipher simply takes the entire original message and shifts everything some number of times to a given direction (wrapping around at the ends). Write a function

```
function ineffectiveCipher(message, shiftNum, direction)
```

which takes the original message **`message`**, the number of times to shift **`shiftNum`**, and `"left"` or `"right"` as a **`direction`** to shift. The function returns the encrypted form of the message after using our ineffective cipher. Note: don't forget the case where `shiftNum > message.length`!

For example:

```
ineffectiveCipher("106J students are awesome!", 3, left)
returns "J students are awesome!106"
ineffectiveCipher("106J students are awesome!", 3, right)
returns " me!106J students are aweso"
```

[1] Shift cipher definition adopted from CS155 lecture1 notes: crypto.stanford.edu/cs155/