`Eric Roberts
CSCI 121

Handout #20
September 26, 2018

Assignment #4—Strings and Characters

Due: Tuesday, October 2, 11:59 P.M. [Note that this assignment is due on Tuesday]

The primary purpose of this assignment is to give you practice working with strings in Python, since that material is covered on the midterm exam on October 3.

Problem 1 (Chapter 6, exercise 1, page 220)

Exercise 14 in Chapter 2 asks you to write a program to find perfect numbers. Solve that exercises but extend it so that it also displays the binary form of these numbers. As you can see if you run this program, the first few perfect numbers follow an interesting pattern when you write them out in binary. Euclid discovered this pattern more than 2000 years ago, and the 18th-century Swiss mathematician Leonhard Euler proved that all even perfect numbers follow this pattern.

Problem 2

As described in Chapter 1, Python allows you to convert a numeric string to an integer using the built-in function `int` and to convert an integer to a string using the built-in function `str`. Suppose that these functions did not exist. Write a Python module called `intstr.py` that export functions `int_to_str` and `str_to_int` that perform these conversions using only the following operations:

- The arithmetic operators
- The control statements described in Chapter 2
- The functions `chr` and `ord`
- String concatenation using the `+` operator

You may assume—as is true for Unicode—that the digit characters are consecutive in the encoding. Thus, if `ch` contains the code for a digit character, the numeric value of that digit is given by `ord(ch) - ord("0")`. For example, if `ch` contains the character `"7"`, the expression `ord(ch) - ord("0")` subtracts the Unicode value for `"0"` (which is 48) from the Unicode value for `"7"` (which is 55) to get the number 7.

Problem 3 (Chapter 6, exercise 6, page 221)

The concept of a palindrome is often extended to full sentences by ignoring punctuation, spacing, and differences in the case of letters. For example, the sentence

Madam, I'm Adam.

is a sentence palindrome, because if you look only at the letters and ignore any case distinctions, it reads identically backward and forward.

Write a predicate function **`isSentencePalindrome(s)`** that returns **`True`** if **`s`** fits this definition of a sentence palindrome. For example, you should be able to use your function to reproduce the following IDLE session:

```
IDLE
>>> from SentencePalindrome import isSentencePalindrome
>>> isSentencePalindrome("Madam, I'm Adam.")
True
>>> isSentencePalindrome("Able was I ere I saw Elba.")
True
>>> isSentencePalindrome("Not a palindrome.")
False
>>>
```

Problem 4 (Chapter 6, exercise 11, page 223)

When large numbers are written 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 as 1,000,000. Implement a function **`addCommas(digits)`** that takes a string of digits representing a number and returns the string formed by inserting commas at every third position, starting on the right. Your implementation of the **`addCommas`** function should be able to reproduce the following IDLE session:

```
IDLE
>>> from AddCommas import addCommas
>>> addCommas("17")
17
>>> addCommas("2001")
2,001
>>> addCommas("12345678")
12,345,678
>>> addCommas("999999999")
999,999,999
>>>
```