Name: (as it would appear on official course roster)
UCSB email address: **@ucsb.edu**
Lab Section:
Optional: name you wish to be called if different from above
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone")

h08: ASCII Characters and Recursion

Assigned: Thursday, May 30th, 2019
Due: Thursday, June 6th, 2019
Points: 100

- You may collaborate on this homework with AT MOST one person, an optional "homework buddy". MAY ONLY BE TURNED IN THE LECTURE LISTED ABOVE AS THE DUE DATE. There is NO MAKEUP for missed assignments; in place of that, we drop the single lowest score (if you a zero, that is the lowest score.)
- **IMPORTANT**: When submitting this homework:
  - **DO NOT USE STAPLES**
  - WRITE YOUR **NAME ON EACH PAGE** IN THE SPACE PROVIDED
  - USE DARK INK PENS – PLEASE **DO NOT USE PENCIL**
  - PRINT THIS HOMEWORK **DOUBLE-SIDED** PLEASE!
- **REMEMBER**: If you use code/techniques we have not learned in class, you will **NOT** get credit!

READING ASSIGNMENT: Read Chapter 6.3 in Perkovic, review your lecture slides/notes. Then complete these problems.

1. (50 pts) Recall the function **`MirrorEncrypt(string)`** from class lecture and that we demonstrated how it works well with lower-case letters, but does not work with upper-case letters. **Modify it** so that it can do the same for upper-case letters as it can with lower-case letters. In other words, I want **`encrypt("ABCDEFGHIJKLMNOPQRSTUVWXYZ")`** to be able to return **`"ZYXWVUTSRQPONMLKJIHGFEDCBA"`** and I want **`encrypt("Penelope")`** to be able to return **`"Kvmvolkv"`**. Assume that these input strings do not have any other characters in them (i.e. just upper-case or lower-case alphabets – nothing else, so no need to check for those). Use the NEXT page to write out your answer.

Name:
(as it would appear on official course roster)

ANSWER TO Q. 1 HERE:

`MirrorEncrypt(string):`

Name:

(as it would appear on official course roster)

2. (50 pts) Examine this recursive function (suggestion: type it in and try it out) and answer the following questions below:

```
def ispal(MyStr):
    if len(MyStr) < 2:
        return True
    if MyStr[0] != MyStr[-1]:
        return False
    else:
        return ispal(MyStr[1:-1])
```

a. (5 pts) What happens when you call it as: **`ispal("treert")`**?

b. (5 pts) What happens when you call it as: **`ispal("madam I'm adam")`**?

c. (5 pts) What happens when you call it as: **`ispal("Madamimadam")`**?

d. (5 pts) What happens when you call it as: **`ispal("3loopypool3")`**?

e. (5 pts) What is this function looking for (in one sentence)?

f. (5 pts) Identify the base case(s) of this recursive function.

Name:

(as it would appear on official course roster)

g. (20 pts) Write a new function **`Check_ispal(word)`** that takes a string input (**word**) and: (a) strips all **non-alphanumeric characters** (that is, all characters that are not lowercase/uppercase letters or numbers), from **word**, then (b) converts all uppercase letters in **word** to lowercase letters and finally, (c) calls **ispal()** (the one defined in the previous page) with the modified word string. So, when called this way:
`Check_ispal("A man, a plan, a canal: Panama‼")`, it returns **`True`**, and
`Check_ispal("A Santa at NASA")`, also returns **`True`**, and
`Check_ispal("Yawn... Madonna fan? No damn way!!")`, also returns **`True`**.

I'm giving you a head-start with a "skeleton code" to complete:

```
def Check_ispal(word):
      print(word, end = ": ")
      newword = ''
      for c in word:
```

Name:

(as it would appear on official course roster)

EXTRA QUESTIONS 4 U!

These are extra problems for you to work on IF you want. These are COMPLETELY OPTIONAL to do and will not be graded. Solutions will be given when the homework is due.

A. What does the function **xyz()** when called, as shown below, do exactly? And what is the end result of this program? Explain each step.

```
def xyz(n):
 if n < 2:
    return 3
 else:
    return 1 + xyz(n – 1)

a = xyz(5)
b = xyz(9)
print(a – b)
```

B. Consider the function **enc()** below:

```
def enc(s):
    result = ''
    for c in s:
        nc = chr(2*ord(c) + 1)
        result += nc
    return result
```

Write a function **dec()** that will decrypt function **enc()** so that if I run:

```
print(dec(enc("hello")))
```

I will see **`hello`** printed back.

Name:

(as it would appear on official course roster)

C. Write a function **custom_enc(message)** that utilizes a user (standard) input string (called a key string) to encode the string **message**.

The function must ask the user to **"Enter your key string (must be less than 10 characters long): "** and it must check to see that this key string is actually less than 10 characters long (if it is not, then it should keep asking until the entered key string length is indeed less than 10 characters long).

Once this is done, the **custom_enc()** function must apply the following formula for encryption of the string **message**: take the ASCII number of the *first* character in the key string, then add it to the ASCII number of the *last* character in the key string, then subtract the *length* of the key string from that – call that result **n**. Then take each character in the message and add the number **n** to its ASCII number. The function should then return that encoded string result.

As a validation of the code you wrote, if you issue the command **print(cust_enc("send pizzas!"))** and then enter the key string: **cheese**, you will end up with the encrypted message: **wirh$tm~~ew%** and if you enter the key string: **1234**, you will end up with the encrypted message **tfoe!qj{{bt"**