Computer Science 1-Honors	Lab 14B Practice/Perform Major Python Assignment
"All Kinds Of Palindromes"	80, 90, 100 & 110 Point Versions

Assignment Purpose:

To gain a deeper understanding of string processing

Write a program that determines if an entered string is a *Palindrome*. True Palindromes are strings of characters that read the same backward as forward. Examples of Palindromes are:

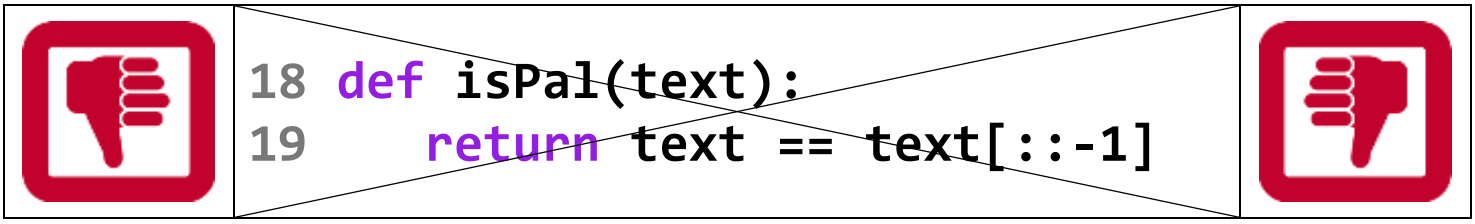
MADAM, RACECAR, BOB, HANNAH, CIVIC, KAYAK, LEVEL, REVIVER

Lab 14B Student Version	Do not copy this file, which is provided.

```
# Lab14Bst.py
# "All Kinds Of Palindromes"
# This is the student, starting version of Lab 14B.
# NOTE: This lab is meant for students in CS1-HONORS.
#       Students in REGULAR CS1 will do Lab 14A.


def heading():
    print()
    print("*********************************************")
    print("Lab 14B, All Kinds Of Palindromes")
    print("80 Point Version")
    print("By: JOHN SMITH") # Substitute your own name here.
    print("*********************************************")


def isPal(text):
    return False


def purge(text):
    return text


def leastPal(text):
    return ""


```

```
30 ##########
31 #  MAIN  #
32 ##########
33
34 heading()
35 finished = False
36 while not finished:
37    print("\n")
38    text = input("Enter a string  -->  ")
39    print("\nPalindrome:         ",isPal(text))
40    print("Almost Palindrome: ",isPal(purge(text)))
41    print("Least Palindrome:  ",leastPal(text))
42    choice = input("\nDo you wish to repeat this program? {Y/N} --> ")
43    if choice.upper()[0] != 'Y':
44       finished = True
45
```

NOTE: There is an advanced form of *String Slicing* that lets you write the **isPal** function with a single command (shown below). This is NOT ALLOWED for this assignment. You will not receive ANY credit for Lab 14B if you code the **isPal** function in this manner.



80 Point Version Specifics

The main thing this program needs to do is determine if an entered string is a *Palindrome*. To do this, you must complete the **isPal** function. Right now, this function has a single line of code, which just returns **False**. This simply allows the program to execute. To complete the **isPal** function, you need to write the necessary code so that it returns **True** if the entered string is a *Palindrome* and returns **False** if it is not.

For this version, the **isPal** function is *case-sensitive* meaning that *madam* and *MADAM* are Palindromes, but *Madam* and *madaM* are not.

You also are not concerned with checking if the String is an *Almost Palindrome* or computing the *Least Palindrome*. The program will generate some output for these, but it can be ignored.

80 Point Version Output

```
 ----jGRASP exec: python Lab14Bv80.py

*******************************************
Lab 14B, All Kinds Of Palindromes
80 Point Version
By: JOHN SMITH
*******************************************


Enter a string  -->  MADAM

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  Bandana

Palindrome:          False
Almost Palindrome:   False
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  RaceCar

Palindrome:          False
Almost Palindrome:   False
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> N

 ----jGRASP: operation complete.
```

For the 80 and 90 Point Versions, *Almost Palindrome* will always have the same result as *Palindrome* and *Least Palindrome* will always be blank.

90 Point Version Specifics and Output

The 90 point version is very similar to the 80 point version except now the **isPal** method is no longer *case sensitive*. So while *madam* and *MADAM* were already Palindromes, now *Madam, mADAM*, and *mADam* are Palindromes as well. You still are not concerned with checking if the String is an *Almost Palindrome* or computing the *Least Palindrome*. The program will generate some output for these, but it can be ignored.

```
  ----jGRASP exec: python Lab14Bv90.py

*****************************************
Lab 14B, All Kinds Of Palindromes
90 Point Version
By: JOHN SMITH
*****************************************


Enter a string  -->  MADAM

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  Bandana

Palindrome:          False
Almost Palindrome:   False
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  RaceCar

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> N

  ----jGRASP: operation complete.
```

For the 80 and 90 Point Versions, *Almost Palindrome* will always have the same result as *Palindrome* and *Least Palindrome* will always be blank.

100 Point Version Specifics

The 100-point version requires everything from the 90-point version and adds the ability to detect *Almost Palindromes.* An *Almost Palindrome* is a sentence or phrase that becomes a Palindrome when you remove any characters that are not letters. Here are a few examples:

Madam I'm Adam. **A man, a plan, a canal, Panama** **Not A Banana Baton!**

You may have noticed that there is no function in the program with aname like **almostPal**. Pay close attention to this line of code from the **MAIN** section of the program:

```
print("Almost Palindrome: ",isPal(purge(text)))
```

What we see here is that 2 functions are called. There is the **isPal** function, which you already wrote for the 90-point version, and then there is the **purge** function. The purpose of this function is to create a new string comprised from only the letters from the original *text* string parameter. Any character that is not a letter is discarded. This new string, which only contains letters, is what is returned by the **purge** function. And if this "purged" string is a Palindrome, then the original string must be an *Almost Palindrome*.

You still are not concerned with computing the *Least Palindrome*. It will simply be blank.

100 Point Version Output

```
 ----jGRASP exec: python Lab14Bv100.py

****************************************
Lab 14B, All Kinds Of Palindromes
100 Point Version
By: JOHN SMITH
****************************************


Enter a string  -->  MADAM

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y
```

For the purpose of this program, any word that is already a *Palindrome*, will also be an *Almost Palindrome*.

```
Enter a string  -->  Bandana

Palindrome:          False
Almost Palindrome:   False
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  RaceCar

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  A man, a plan, a canal, Panama

Palindrome:          False
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  +-*/===

Palindrome:          False
Almost Palindrome:   True
Least Palindrome:

Do you wish to repeat this program? {Y/N} --> N

  ----jGRASP: operation complete.
```

Just like the 90-point version, the 100-point version is not case-sensitive.

Since this 5th input has no letters whatsoever, it is guaranteed to be an *Almost Palindrome*. When all non-letter characters are removed, there is nothing left, and an empty string is a *Palindrome*.

110 Point Version Specifics

The 110 point version requires everything from the 100 point version and adds the ability to create *Least Palindromes*. Consider the word *Banana*. It is not a *Palindrome*. It is also not an *Almost Palindrome*. However, any string can be made into a *Palindrome* if you reverse the string and then concatenate it to the end of the string. So in the case of *Banana*, the reverse is *ananaB*. If we concatenate these we get *BananaananaB* which is a *Palindrome*, but it is not the *Least Palindrome*. The reason it is not the *Least Palindrome* is we concatenated more characters than are necessary. All we need is to concatenate a *B* to the end and we get *BananaB,* which not only is a *Palindrome*, it is the *Least Palindrome*. By definition, this means if the entered string was already a *Palindrome*, like *MADAM*, then is already is a *Least Palindrome*. In other words, the *Least Palindrome* of *MADAM* is *MADAM*. One more example is *Panama*. Simply by concatenating *naP* we get *PanamanaP*.

110 Point Version Output

```
 ----jGRASP exec: python Lab14Bv110.py

*****************************************
Lab 14B, All Kinds Of Palindromes
110 Point Version
By: JOHN SMITH
*****************************************


Enter a string  -->  MADAM

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:    MADAM

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  Bandana

Palindrome:          False
Almost Palindrome:   False
Least Palindrome:    BandanadnaB

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  RaceCar

Palindrome:          True
Almost Palindrome:   True
Least Palindrome:    RaceCar
```

Since the 1st & 3rd strings (*MADAM* and *RaceCar)* are Palindromes, their *Least Palindromes* will be exactly the same.

```
Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  A man, a plan, a canal, Panama

Palindrome:          False
Almost Palindrome:   True
Least Palindrome:    A man, a plan, a canal, PanamanaP ,lanac a ,nalp a ,nam A

Do you wish to repeat this program? {Y/N} --> Y


Enter a string  -->  +-*/===

Palindrome:          False
Almost Palindrome:   True
Least Palindrome:    +-*/===/*-+

Do you wish to repeat this program? {Y/N} --> N

 ----jGRASP: operation complete.
```