Midterm

Total 600 pts. 10 questions.

Question 1: 60 pts. Implement a function with a set of test code that converts from feet to inches. Call the file `feet_to_in.py`. There are 12 inches to the foot.

Question 2: 60 pts. Use a loop to search a list and find if a value is in the list. Implement a function, `find_in_list` that takes 2 parameters.

```
def find_in_list ( theList, lookFor ):
```

and returns True if the value in `lookFor` is in the list in `theList`.

Implement a simple test for the function, `find_in_list`. Put the code in the file `find_in_list.py`.

An example of calling the function (can be used in your test) is:

```
ll = [ "a", "b", "c" ]
found = find_in_list ( ll, "b" )
```

Should return true,

and

```
ll = [ "a", "b", "c" ]
found = find_in_list ( ll, "x" )
```

should return false.

Question 3: 60 pts. Implement a function to convert an input list of strings to a single string. (This is covered in Lecture 10 - there is an example) Have a set of test code for it.

Question 4: 60 pts. Use Python (either write a very short program, or use it interactively) to calculate:

```
y = k + 4 * x  + 2 * x * x
```

where k = 92827828272222

and x = 4893849328949900000993939393939394782920202092002020207272224343 3

Turn in the resulting 'y' value in a text file, called `question4.md` . Print out `y` .

Question 5: 60 pts. You want to find strings where the first 3 characters are the reverse of the last 3 characters in the following list of strings:

```
ll = [
        "abc earth cba",
        "abc mars abc",
        "abcba"
]
```

You have a function that will reverse strings:

```
 1:
 2: # revers will reverse a string, putting the last character first etc.
 3: def reverse(s):
 4:     i = len(s)-1
 5:     o = ""
 6:     while ( i >= 0 ):
 7:         o = o + s[i]
 8:         i = i - 1
 9:     return ( o )
10:

11:
12: # Automated Test
13: if __name__ == "__main__":
14:     n_err = 0
15:     x = reverse ( "The quick brown fox jumps over the lazy dog." )
16:     expect = ".god yzal eht revo spmuj xof nworb kciuq ehT"
17:     if x != expect:
18:         n_err = n_err + 1
19:         print ( "Error: Test 1: conversion not working, expected {} got {}".format (  expect, x ) )
20:     x = reverse ( "" )
21:     if x != "":
22:         n_err = n_err + 1
23:         print ( "Error: Test 2: conversion not working, expected ->{}<- got ->{}<-".format ( "", x ) )
24:
25:     if n_err == 0 :
26:         print ( "PASS" )
27:     else:
28:         print ( "FAILED" )
29:
```

Write a program that will take the input list above and find the strings that have the first 3 character matching the reverse of the last 3 characters.

Question 6: 60 pts. You have a table

age bracket	expected value
0 to 14	0
15 to 16	2000
17 to 18	6000
19 to 23	24000
24 to 40	38200
41 to 67	51000
67 and older	18200

Implement this as a function with if/elif/else and return the value from the 2nd column. Implement a test that checks this.

Question 7: 60 pts. Given the following code:

```
1: ll = [ "a", "b", "c", "d" ]
2: for i in range(len(ll) ):
3:     print ( "i={} ll[{}]= ->{}<-".format ( i, i, ll[i] ) )
4: print ( "that's all folks...." )
```

Show the output from the 4 line program. (Take the code and run it and save the output to a .md file)

Question 8: 60 pts. Using the following code

```
1: s = "abc"
2: i = len(s)-1
3: o = ""
4: while ( i >= 0 ):
5:     o = o + s[i]
6:     i = i - 1
7: print ( "o = {}".format(o) )
```

Hand trace what is happening in this code. Complete the following table.

Line No / Time	s	i	o	notes
1l / t1	abc			
l2 / t2	abc	2		
l3 / t3	abc	2	""	
l4 / t4	abc	2	""	while is true, enter loop
l5 / t5	abc	2	"c	

You should have aroud 14 steps - the blanks in the table are values that do not exist. You add additional steps to the bottom of the table. You can do this by hand or run the code and add print statements. You can put the table in excel or numbers. You can build a little table in a text file or in a markdown file.

Question 9: 60 pts. You have the following code. It is not working correctly. Fix it. (file is q9.py) There are 3 yards to 1 foot. Determine if the function feet_to_yards is returning the correct value or if the test case is incorrect.

```
 1:
 2: # Convert from feet to yards
 3: def feet_to_yards ( feet ):
 4:     conv = 3
 5:     yards = feet * conv
 6:     return (feet)
 7:
 8: # Automated Test
 9: if __name__ == "__main__":
10:     n_err = 0
11:     x = feet_to_yards ( 3 )
12:     if x !=  1:
13:         n_err = n_err + 1
14:         print ( "Error: Test 1: conversion not working, expected {} got {}".format (  1, x ) )
15:     x = feet_to_yards ( 0 )
16:     if x != 0:
17:         n_err = n_err + 1
18:         print ( "Error: Test 2: conversion not working, expected {} got {}".format ( 0, x ) )
19:
20:     if n_err == 0 :
21:         print ( "PASS" )
22:     else:
23:         print ( "FAILED" )
24:
```

Question 10: 60 pts. You have the following code. It is not working correctly. Fix it. (file is q10.py)

Hint: you may want to fix the code before you run it - or remember that a control-C will terminate a program in the middle of running it

The code is supposed to take a list of values and double each of the values in in a loop.

```
 1:
 2: def double_values_in_list ( ll ):
 3:     i = 0
 4:     while ( i < len(ll) ):
 5:         ll[i] = ll[i] * 2
 6:         print ( "ll[{}] = {}".format( i, ll[i] ) )
 7:
 8:     return ll
 9:
10:
11:
12: # Automated Test
13: if __name__ == "__main__":
14:     n_err = 0
15:     ll = [ 1, 2, 3 ]
16:     x = double_values_in_list ( ll )
17:     if x[0] != 2:
18:         n_err = n_err + 1
19:         print ( "Error: Test 1: conversion not working, expected {} got {}".format (  2, x[0] ) )
20:
21:     if n_err == 0 :
22:         print ( "PASS" )
23:     else:
24:         print ( "FAILED" )
25:
```