**COS 120 - Introduction to Computational Problem Solving**

**Lab 06 - Lists**

Show the completed code for each of these exercises to your TA, as you complete each one. You are responsible to be sure the TA has checked off completed exercises for this lab. Unfinished exercises may be completed and turned in by midnight today.

**L06-1)** Writea function **def showListFunctions(myClasses):**  Pass this function a list containing the class designations for each of the classes you are taking this semester (e.g. "COS 120", "ENG 110", etc.). Test your comprehension of list processing by printing output from the following operators/functions (common also to strings) on your list (be creative): (7 points)

a) slicing

b) membership (in and not in)

c) concatenation

d) indexing (using brackets [ ])

e) length

f ) repetition

g) index (method)

**>>> cls=["COS 120", "ENG 110", "SYS 100", "BIB 110"]**

**>>> showListFunctions(cls)**

**['SYS 100', 'BIB 110']**

**True**

**False**

**['COS 120', 'ENG 110', 'SYS 100', 'BIB 110', 'SYS 393']**

**COS 120**

**4**

**['COS 120', 'ENG 110', 'SYS 100', 'BIB 110', 'COS 120', 'ENG 110', 'SYS 100', 'BIB 110']**

**ENG 110 found at index 1**

**>>>**

**L06-2)** Write a function **def showTimeForClass(classes, times):**  that takes two lists as arguments**.** Pass **t**his function the same list described above in problem L05-1, and pass a second list of the time (in military designation) each class begins (e.g. 800 for 8:00am, 1300 for 1:00pm). Of course, the time list should be in the same order as the class list (these are parallel lists!). The function should prompt the user to enter a class designation (e.g. COS 120). Display the corresponding start time for that class. (10 points)

**>>> myClasses=["COS 120", "ENG 110", "SYS 100", "BIB 110"]**

**>>> classTimes=[1100,800,1300,1500]**

**>>> showTimeForClass(myClasses,classTimes)**

**Enter a course designation => SYS 100**

**1300**

**L06-3)** Append the time list onto the class designation list from problem L05-2 (in other words create a **single list** with all of the course designations, followed by all of the start times). Rewrite the function from L06-2 (**call it showTimeForClass2**) to allow the user to specify a class designation (e.g. "COS 120"), and return the corresponding start time as before, but remember that the start time is now found in the same list. (The order is important here, too!) (10 points)

**>>> myClasses=["COS 120", "ENG 110", "SYS 100", "BIB 110"]**

**>>> classTimes=[1100,800,1300,1500]**

**>>> combined=myClasses+classTimes**

**>>> showTimeForClass2(combined)**

**Enter a course designation => SYS 100**

**1300**

**L06-4)** Write a function **def demoListMethods(aList):**  to demonstrate the use of the following methods on a list of your choice (perhaps one of the above lists?): (8 points)

a) append

b) insert

c) pop (both forms)

d) sort

e) reverse

f) index

g) count

h) remove

**>>> cls=["COS 120", "ENG 110", "SYS 100", "BIB 110"]**

**>>> demoListMethods(cls)**

**['COS 120', 'ENG 110', 'SYS 100', 'BIB 110', 'IAS 110']**

**['SYS 492', 'COS 120', 'ENG 110', 'SYS 100', 'BIB 110', 'IAS 110']**

**['SYS 492', 'COS 120', 'ENG 110', 'SYS 100', 'BIB 110']**

**['COS 120', 'ENG 110', 'SYS 100', 'BIB 110']**

**['BIB 110', 'COS 120', 'ENG 110', 'SYS 100']**

**['SYS 100', 'ENG 110', 'COS 120', 'BIB 110']**

**0**

**1**

**['ENG 110', 'COS 120', 'BIB 110']**

**L06-5)** Using list methods or functions **with the exception of .reverse (you may not use .reverse)**, write three forms of a function **def reverseList(aList):** which takes a list as an argument and returns a new list in reverse order. (9 points)

**>>> reverseList([1,2,3,4,5])**

**[5, 4, 3, 2, 1]**

**L06-6)** Write a function def shuffleToNewList(alist) that accepts a list as a parameter, and randomly shuffles its contents into a new list which it returns. Do NOT use .shuffle

**L06-7)** Write a function def shuffleInList(alist) that accepts a list as a parameter, and randomly shuffles its contents in the original list. Do NOT use .shuffle