**Q.1**  **Dictionary Merge**

Write a program to merge two dictionaries into one.

**Q.2**  **Duplicate Elimination**

Create a function that receives a list and returns a (possibly shorter) list containing only the unique values in sorted order. Test your function with a list of numbers and a list of strings.

**Q.3**

Define the **getWordRangeDict()** function which is passed a string as a parameter. The parameter string is made up of words, each followed by a number (the same word may occur more than once in the string). The function creates a dictionary where the keys are all the unique words from the string and the corresponding values are tuples made up of two numbers, the smallest number which follows the word and the largest number which follows the word. If there is only one occurrence of a particular word then the smallest number and the largest number will be the same number.

Note: the testing code makes use of the **printDictInKeyOrder(aDict)** which prints the dictionary pairs in sorted key order. You also need to define this function.

For example, the following code:

**aDict = getWordRangeDict("cat 4 dog 3 dog 2")**

**printDictInKeyOrder(aDict)**

**print()**

**sentence = """cat 4 dog 3 dog 2 elephant 7 fish 5 elephant 6 fish 5   
 dog 6 elephant 9 fish 15 dog 4 dog 9 elephant 1"""**

**aDict = getWordRangeDict(sentence)**

**printDictInKeyOrder(aDict)**

outputs:

**cat : (4, 4)**

**dog : (2, 3)**

**cat : (4, 4)**

**dog : (2, 9)**

**elephant : (1, 9)**

**fish : (5, 15)**

**Q.4 Is a Sequence Sorted?**

Create a function **is\_ordered** that receives a sequence and returns True if the elements are in sorted order. Test your function with sorted and unsorted lists, tuples and strings.

**Q.5 Right Triangle**

Write a function that takes three integers as its input and returns whether a triangle with those side lengths is a right triangle or not. Hint: sort is your friend.

**Q.6**

Given a list of items, write a program that generates a list of lists of the following form:

**[a,b,c,...,z] ⇒ [[z], [y,z], [x,y,z], ... , [a,b, ... ,y,z]]**

Hint: Slicing is your friend.

**Q.7 Hexadecimal to Decimal Conversion**

Write a program that converts a hexadecimal number into a decimal number without using the **int** function. The program will prompt the user to enter a hex number as a string and convert it into a decimal. For example, for the input hex number AB8C, the output from your program should be 43916.

**Q.8 Pattern Recognition: four consecutive equal numbers**

Write the following function that tests whether the list has four consecutive numbers with the same value:

**def isConsecutiveFour(values):**

Write a test program that prompts the user to enter a series of integers and reports whether the series contains four consecutive numbers with the same value.

**Q.9 Cipher**

A Caesar cipher encrypts a message by shifting letters in the alphabet. For example, a shift of 4 maps 'a' to 'e' and maps 'p' to 't' Here is a famous line from Shakespeare encrypted with a shift of 4: 'vq dg qt pqv vq dg: vjcv ku vjg swguvkqp.'

1. Write a program that takes as input a string to be encrypted and an integer encryption shift (such as 4 mentioned earlier), and prints the encrypted string. [Hint: zip() is helpful in building a dictionary. Also, remember to handle space – it doesn't shift].
2. Extend your program to take an additional input that indicates if your program is to encrypt or decrypt the string.

**Q.10 Tic-Tac-Toe (aka Noughts and Crosses)**

Implement the Tic-Tac-Toe (aka Noughts and Crosses) game. Use the **random** library to choose a place on the board and put the mark. Display the board after each turn. The game ends when there is a winner or a tie.

You don't need fancy graphics (Qt-based, etc.) for board display but that is entirely up to you. The following plain-text output works just fine:

| X | O | O |

| X | O | O |

| X | O | O |

**Q.11 Recursion**

A rabbit can move forward by taking one or two steps at a time. Write a recursive program to enumerate all possible ways a rabbit can move forward ***n*** steps.

To wit, if ***n*** is 3, the program should output the following:

**[ 1 1 1 ]**

**[ 1 2 ]**

**[ 2 1 ]**