Programming Assignment 6

1. Write a function, called **listToDict**, that takes a list as an argument and returns a dictionary containing the keys 1 through n, where n is the size of the list, and the values correspond to the values in the list.
   For example, if the list is [2, 6, 6, 1, 7, 9] then the dictionary maps 1 —> 2, 2 —> 6, 3 —>6, 4 —> 1, 5 —> 7 and 6 —> 9. (20 points)

2. Write a function, called **newDict**, that takes a dictionary as an argument and returns a new dictionary with the keys and values inverted (keys become values and values become keys).
   What happens if there are duplicate values? Short comment with the answer if fine (20 points)

3. Write a function, called **uniqueElems**, that takes a list of values as a parameter and determines if all elements are unique (no repeated values). Return True if all values are unique, False otherwise. Think of a way to use dictionary to perform this task. (20 points)

4. Write a function, called **valFrequency**, that given a list of values as a parameter, counts the frequencies for each value in the list. You can do this by returning a dictionary (think about what the key should be and what value should be associated with it).
   For example, if the list is [1, 3, 5, 2, 1, 2, 5, 8, 4, 5] then we have 2 x1's, 1 x 3's, 3 x 5's, 2 x 2's, 1x4's, 1x8's (20 points)

5. Write a function, called **addsToK**, that given an integer k and a list of n unordered integers A, determines if there is a pair of **distinct** integers in A that add up to k. Returns True if they are, False otherwise.
   For example : given [1, 6, 7, 3, 7, 10, 3] if k=13 then there is a pair of integers that add up to 13 : 10 and 3. If k=14 then there isn't a pair of distinct integers that add up to 14 (can't use 7 twice even if it appears twice in the list) (20 points)

-Please remember that all functions need to have proper DocString documentation.
-Refer to the CE_functionComment0.pdf and CE_functionComment1.pdf files for reference.
-Please avoid the use of global variables in your code. Any variable that is required for your functions to perform their respective tasks must be defined inside of the functions themselves. Only the input to your functions, when it applies, can be defined in the global scope.