Homework 2

Math 510

All functions and documentation should be in a single file titled `lastname_firstname_HW02.py` (not an .ipynb file) where your last and first name are as they appear on your student ID. The homework is to be completed in Python 3.x. All functions in the homework should be titled and take input as stated in the problem specification.

Your function will only be tested with the input types described in the problem specification. When possible try and reuse functions from earlier questions (as code is intended to be reusable). Pay attention if the specification asks for a **return** or a **print**. Remember to document your code (purpose, input, output), as this will be an important part of your grade. Good luck.

1. For both parts be careful to make sure that you don't change the values of the original list being passed to the function.

- Write a function, **has_duplicate(input_list)**, that accepts a list and returns True if the list has a duplicate item and False if it does not.
- Write a function, **remove_duplicate(input_list)**, that accepts a list and returns the list, in order, keeping only the first instance of any duplicated values.

2. Write a function, **invert_dict(input_dictionary)**, that accepts a dictionary whose keys and values are both of immutalbe types, and returns a dictionary whose keys are the values of the input dictionary and whose values are ordered lists of keys that share that value.

3. A caeser cipher is when you take a string and shift each letter three to the left i.e. 'abcde' becomes 'xyzab'. Write a function, **caesar_cipher(input_text)**, that accepts a string (possibly containing spaces and punctuation) and returns the lower cased encrypted string. Characters not from the alphabet, such as spaces and punctuation, should remain unchanged.

4. Write a function, **make_dictionary(file_name)**, that accepts a file name and returns a dictionary where the key is a letter and the value is the set of words from the file that begin with that letter. Assume that the file contains one word per line.

5. Write a function **anagrams(file_name)** that accepts a file_name that contains a list of words (one per row) and returns the larget set of anagrams in the word list. i.e. art,tar,rat would be a set of 3. If there is a tie it should return one of the sets. First try running it on test.txt, but it should run in a reasonable amount of time (<3m) if using unixdict.txt
HINT: Look at the code from class for checking if two words are anagrams.