LAB 3

1.Write a function that receives as parameters two lists a and b and returns a list of sets containing: (a intersected with b, a reunited with b, a - b, b - a)

2. Write a function that receives a string as a parameter and returns a dictionary in which the keys are the characters in the character string and the values are the number of occurrences of that character in the given text.

Example: For string "Ana has apples." given as a parameter the function will return the dictionary: {'a': 3, 's': 2, '.': 1, 'e': 1, 'h': 1, 'l': 1, 'p': 2, ' ': 2, 'A': 1, 'n': 1} .

3. Compare two dictionaries without using the operator "==" and return a list of differences as follows: (Attention, dictionaries must be recursively covered because they can contain other containers, such as dictionaries, lists, sets, etc.)

4. The build_xml_element function receives the following parameters: tag, content, and key-value elements given as name-parameters. Build and return a string that represents the corresponding XML element. Example: build_xml_element ("a", "Hello there", href =" http://python.org ", _class =" my-link ", id= " someid ") returns the string = " Hello there "

5. The validate_dict function that receives as a parameter a set of tuples (that represents validation rules for a dictionary that has strings as keys and values) and a dictionary. A rule is defined as follows: (key, "prefix", "middle", "suffix"). A value is considered valid if it starts with "prefix", "middle" is inside the value (not at the beginning or end) and ends with "suffix". The function will return True if the given dictionary matches all the rules, False otherwise.

Example: the rules s={("key1", "", "inside", ""), ("key2", "start", "middle", "winter")} and d= {"key1": "come inside, it's too cold out", "key3": "this is not valid"} => False because although the rules are respected for "key1" and "key2" "key3" that does not appear in the rules.

6. Write a function that receives as a parameter a list and returns a tuple (a, b), representing the number of unique elements in the list, and b representing the number of duplicate elements in the list (use sets to achieve this objective).

7. Write a function that receives a variable number of sets and returns a dictionary with the following operations from all sets two by two: reunion, intersection, a-b, b-a. The key will have the following form: "a op b", where a and b are two sets, and op is the applied operator: |, &, -.

Ex: {1,2}, {2, 3} =>

{

"{1, 2} | {2, 3}": 3,

"{1, 2} & {2, 3}": 1,

"{1, 2} - {2, 3}": 1,

...

}

8. Write a function that receives a single dict parameter named mapping. This dictionary always contains a string key "start". Starting with the value of this key you must obtain a list of objects by iterating over mapping in the following way: the value of the current key is the key for the next value, until you find a loop (a key that was visited before). The function must return the list of objects obtained as previously described.

Ex: loop({'start': 'a', 'b': 'a', 'a': '6', '6': 'z', 'x': '2', 'z': '2', '2': '2', 'y': 'start'}) will return ['a', '6', 'z', '2']

9. Write a function that receives a variable number of positional arguments and a variable number of keyword arguments adn will return the number of positional arguments whose values can be found among keyword arguments values.

Ex: my_function(1, 2, 3, 4, x=1, y=2, z=3, w=5) will return returna 3