Laboratory 13: Tuples and Sets

(1) Write a function that creates a random set of *n* elements (*n* being the sole parameter of the function). Each element is a tuple of two coordinates, (x, y), where x is a random number between 0 and 9 and y a random number between 0 and 999.

(2) Write a function that takes a set of tuples as input. The function returns the same set, but tuples with the same first coordinate as a previous tuple in the set are excluded. (Hint: There are many possible ways of doing this, some involving even more advanced data structures. The easiest solution to my mind is to iterate through the set, collect the first coordinates of the tuples encountered in a list or set, check whether the first coordinate of a new tuple to be processed is already there, and only add the new tuple to the result set accordingly.) Since Python does not guarantees the ordering of elements in the set, the result is a bit unpredictable.

For an example, I use the function in (1) to create a set of 50 elements.

```
>>> myset = create_random_set(50)
>>> myset
{(8, 601), (6, 42), (5, 946), (2, 212), (0, 811), (9, 632), (4, 801),
(4, 0), (7, 404), (5, 451), (1, 537), (1, 588), (9, 399), (4, 677),
(8, 223), (6, 665), (8, 968), (5, 559), (5, 47), (1, 540), (9, 562),
(3, 186), (0, 296), (3, 196), (4, 897), (7, 610), (2, 378), (5, 766),
(8, 686), (4, 284), (7, 725), (5, 938), (4, 243), (8, 502), (6, 33),
(5, 845), (7, 345), (0, 458), (0, 169), (0, 246), (9, 141), (1, 662),
(9, 547), (8, 958), (5, 20), (3, 617), (1, 550), (6, 79), (3, 91), (5,
99)}
```

I then run the function from (2).

```
>>> myseta = remove(myset)
>>> myseta
{(8, 601), (7, 404), (1, 537), (6, 42), (5, 946), (2, 212), (0, 811),
(3, 186), (9, 632), (4, 801)}
```

As you can see, only 10 tuples survived the selection.

(3) Write a function that removes duplicate elements from a list by converting first the list to a set and then back to a list.

(4) You can create a set of letters in a word by using the set constructor as in `set('ahmedabad')`, which yields `{'e', 'b', 'd', 'h', 'a', 'm'}`. Write a function that checks whether the letters used in the two words given to it as an argument are the same or not. (Two check the equality of two sets, just use the comparison operator `==` .)