Laboratory 8

This laboratory focuses on using threads to resolve some concurrency problems. During the lecture, we have learned that critical sections must be protected by lock.acquire() and lock.release() pieces of code. However, using acquire() and release() method, we must assure that even in the worst situation, if an error or an exception occurs after acquiring the lock, the lock will be released. So, the best approach is

```
try:
    lock.acquire()
    # do the work in critical section
finally:
    lock.release()
```

An alternative approach is to use context managers, exactly as we've already used when working with files. Remember that when we deal with files we do:

```
with open(file_name, "r"):
    # do_something()  with the file
```

In this case, explicitly closing the file is no more necessary, the interpreter is doing this when leave the block defined by `with` clause.

In the same way we can deal with threading locks:

```
with lock:
    # do_something()  inside the critical section
```

When the program flow exits the with block, the `lock.release()` is called.

Problem 1

Given the lists below, create a class derived from `threading.Thread` base class specialized in sorting a list. Use a queue of lists to extract a list and sort it. Print the results.

```
list1 = [5, 2, 7, 1, 9, 9]
list2 = [6, 4, 6, 1, 9, 3, 5, 9, 4]
list3 = [8, 2, 4, 1, 7, 4, 8, 0]
```

Problem 2

Use the same text files in laboratory 5 that define the faces of a dice and create a Dice class derived from `threading.Thread` base class to create a program that will roll dices in parallel. Use as many threads you want to obtain different outputs. (Note that in laboratory 5 we only dealt with two dices).

Problem 3

Create a thread class derived from threading.Thread which is initialized with a file name and a char and it has to count how many time the given char occurs in the given file. In the main program, start many threads pointing to the same file `text.txt` (attached as usually in the Laboratories section) and give different characters to be counted.

Problem 4

Create two classes as follows:

`CardPackage` which models a poker cards package, based on the following values:

```
card_values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'A', 'J', 'Q', 'K']
colors = ["heart", "diamond", "spade", "club"]
```

`PokerPlayer` derived from `threading.Thread`, which models a poker player.

In the main program instantiate four `PokerPlayer` objects and give every player five cards from the package

Use a queue structure to deal the cards. Also, use the `random.shuffle()` method to shuffle the cards in the package, before giving them to the player.

In order to simplify the program, we consider that every player is able to take 5 cards from himself (use a `get_cards()` method in `PokerPlayer` class)

At the end, display what cards every player has.