Lab 2

ABSTRACT DATA TYPES AND IMPLEMENTATIONS

Problem 1: Set implementation as DICT

- Download the file set_impl_as_dict.py and complete all methods that were not implemented yet
- Download the three tests from set_tests.py
- Make sure all three tests in set_tests.py pass after your changes
- Run the new test (test4) that you wrote in problem 1, and make sure it also passes here

Problem 2: Set implementation as LIST

- Download the file set_impl_as_list.py and complete all methods that were not implemented yet
- Download the three tests from set_tests.py
- Make sure all three tests in set_tests.py pass after your changes
- Write a new test (test4) for covering the methods that you have implemented

Problem 3: Which one is Faster?

- We want to write a test that compares the run time performance of our Set implementations
- Download the file set_bench.py and try to understand the code there
- Run this file and try to explain the run time differences there
- Write a similar tests for the **intersection** and **issubset** methods

Problem 4: Bounded Stack

- Write a class Bstack that implements a bounded stack data structure
- A bounded stack is a usual stack that cannot grow above a given limit
- The constructor should accept the limit as an argument:

```
class BStack:
    def __init__(self, limit):
        ...
```

- Write a simple test for checking that the limit works
- If you have copied the Stack code, then you made a code duplication (which is considered bad). Can it be done without code duplication?

Problem 5: MultiSet Implementation

- Write a simple implementation for the MultiSet ADT which is based on the dictionary data structure
- Write a test that covers all the methods of the MultiSet ADT
- Use the MultiSet class to parse all the words in the Oliver Twist book
  - That is, create a MultiSet **mset** which maps every legal word in the book to the number of times it appears in the book
- Write a function `most_common(mset, n)` which accepts a MultiSet `mset` and an integer `n` and returns the elements with the `n` top number of occurrences