CSE 411: Assignment 0
Evaluating Set Implementations (Ungraded)
Due date: Friday September 3 (individuals)

- The goal of this assignment is to compare different C++ implementations of an integer Set interface.

- Experiment behavior: each experiment should perform i operations, randomly chosen among (insert, lookup, remove), each using a random key in the range [0, k-1]

- Semantics: Unlike multisets, duplicates are not allowed in sets. Unlike Maps, there is no value attached with each key (i.e., there is no modify operation). Accordingly, the following three operations should be our API:
  - Insert: returns false if the key is already in the set (we call it unsuccessful), otherwise it inserts the key and returns true (we call it successful).
  - Remove: returns false if the key is not in the set (we call it unsuccessful), otherwise it removes the key and returns true (we call it successful).
  - Lookup: returns true if the key exists in the set (we call it successful), and false otherwise, without changing the data structure (we call it unsuccessful).

- Competitors: compare four implementations (tree, list, sorted vector, unsorted vector)
  - Don't implement them from scratch. start with C++ std containers (http://www.cplusplus.com/reference/stl/)
  - However, make sure you implement the proper adapter (e.g., some containers allow duplicates, have different signatures, are unsorted, … etc).
- Command-line parameters:

  -i: # iterations (int)

  -k: max key (int) (minimum is 0)

  -d: data structure (string) (tree, list, sorted vector, unsorted vector)

  -r: read-only ratio (int) (0-100)

  Note: writes will be evenly split between insert and remove.
- Experiment Setup
  - Pre-initialize to be 50% full. Ideally, the selected keys should be randomly selected. However, a relaxed approach of pre-initializing the set with all "even" keys in the range is acceptable.
  - Measure the time of each experiment using C++ code, not Unix command line, and report the average/variance of 5 runs
  - Be sure to use a big enough i to be valid (1M operations is the minimum)

- Your test should show results for:
  - Different sizes of the Set (e.g., 1K, 10K, 100K, 1M)
  - Different read-only ratios (e.g., 0%, 20%, 50%, 80%, 100%)
- Present your results using clear plots (e.g., varying set size in the x-axis).
- Optional: in addition to the execution time, you can also report the number of successful and unsuccessful operations of each op type (this is important only to make sure that the randomization actually works!!).

- What should be in your report
  - Brief summary of each competitor: only include important details that distinguish each one.
  - Experiment Setup:
    - Fixed parameters (number of iterations, number of trials, …)
    - Tested configurations (sizes, percentages of reads, ...)
  - Plots: showing the results for all configurations you tested.
  - Analysis: should be your "own" analysis of your "own" results, not a general discussion of the implemented data structures. Your analysis should include points like:
    - Which configuration fits each competitor more? why?
    - Is there any unexpected behavior?
    - Is there any flipping points (where a competitor starts to lose against the others instead of winning)? and what is their cause?
  - Don’t copy your source code!! The report is not meant for that!