Data structures project:

Multi Sets, with hash table implementation. For this each element in the array will keep a vector of strings(or whatever type desired)

Set Class:

Variables:

std::vector<item<T>\*> \*hash\_table\_ - an array storing the hash data. Where T is any class.

Functions:

Bool has() this will return if the item exists. Might want an overload that returns the index

Void output\_vals() This will output all the values in the set.

Bool add(T) check if the set has item if not we add it with O(1 + a) time

remove() check if the item exists and if it does we remove it with O(1) time

Int get\_size() return size of set

union(set) we could also just use + operator. Uses add() function O(n)

Intersection(set) takes in two sets and returns a set with only the shared values

subtraction(set)

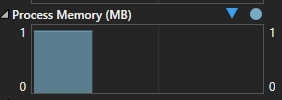
Goal:

Time complexity, and general comparison of different data structures.

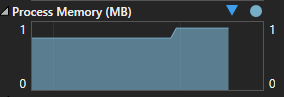
* Arrays
  + Read in hunger games and hamlet ignoring words to ignore. Output number of words and top 10.
  + Record time to complete
* Sets
  + Implement Union, intersection, and Subtraction to between analyze the books similarity and differences.
  + Compare hunger games with words to ignore, and hamlet
  + Compare time when hash table size is changed from ~6000 to 7
* Treaps
  + Uses balanced tree to sort words, implement with any type
  + Output total number of words and specific word
  + Record time.
* Trie trees.
  + Implement simple trie for strings
  + Compare time for reading book and outputting word, and total words
  + Compare data usage

* Compare each data structure when reading in large text files such as hamlet or the hunger games

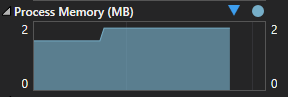
Array



+ Sets



+Treap



+Trie Tree

