Department of Electrical and Computer Engineering
EECE 2560: Fundamentals of Engineering Algorithms

Project #3

Part b

In this part, evaluate the effectiveness of sorting algorithms based on heaps and hashtables.

1. Implement the template class `Heap<T>` that stores objects in a heap of type `vector<T>`, and which includes:

   (a) functions `Parent(int)`, `Left(int)`, `Right(int)`, and `GetItem(int n)` which returns the `n`th item in the heap.

   (b) for a max-heap, functions `InitializeMaxHeap()`, `MaxHeapify()`, and `BuildMaxHeap()`.

   (c) the equivalent functions for a min-heap. You can implement the max-heap and min-heap data structures within the same class and convert the stored data into a min- or max-heap by calling the appropriate member functions.

   (d) function `Heapsort()`

   Since the heap is only used to sort the word list: You can declare the heap within the `WordList::Heapsort` function, copy the unsorted words into the heap, sort the words, and then copy the words out. Then the `WordList::BinarySearch` function can be used to look up words.

2. Implement the template class `HashTable<T>` that stores objects in a hash table of type `vector<vector<T>>`, and which includes:

   (a) functions `AddItem()`, `DeleteItem()`, and `InList()`

   (b) function `Hash()` which returns the hash value for an item

   Since the Hash Table will be used to look up words, you can copy the unsorted words into the hash table, and then the `HashTable::InList` function can be used to look up words. You can make `FindMatches` a template function that can be passed data structures of different types.

3. Measure the runtimes of word search using all the sorting algorithms for all the sample grids. Write a short paragraph summarizing the algorithms' relative performance and submit it with your project.