ASST 0

Emily Tronolone - emt104
Alyssa Santos - ats161
Systems Programming, Spring 2020

Synopsis

This program is designed to read in data from ASCII character files, whether they consist of number or string data, and then sort and output the data. The sorted data outputs the sorted data in increasing order, with every element of data on its own line.

Implementation

1. **Main**

First, the main method checked if the command line arguments were valid. Once the proper files were read in, the number of non-space characters and commas were accounted for. The tokens were then put into a two-dimensional array so the proper sorting functions can be applied. In order to sort the tokens, the program's comparator function would typecast the arguments whether it was numeric data or string data.

2. **Comparator:**

```
int compareFunc(void* ptr1, void* ptr2)
```

The comparator function type casts the function pointer, then checks if the data is a number or string. Then it compares values.

3. **Insertion Sort:**

```
int insertionSort(void* toSort, int (*comparator)(void*, void*))
```

The insertion sort method that's implemented, it takes the array and sorted its elements by evaluating every element in the array and moving it based on its value. This was to ensure that the values would be ordered in ascending order.

4. **Quick Sort:**

```
int quickSort(void* toSort, int (*comparator)(void*, void*))
```

The quick sort method called another quick sort that is implemented in the program that acts as the main quick sort function:

```
void quickSortPt2(char** sortThis, int left, int right, int(*comparator)(void*, void*))
```

In this recursive function, the program uses the partition method, which is traversing from the leftmost element and finding the smaller element to swap with the current element in the array.

5. **Partition:**

```
int partition(char** sortThis, int left, int right, int(*comparator)(void*, void*))
```

This function helped support the quick sort function by traversing from the leftmost element and finding the smaller element to swap with the current element in the array.