Prerak Patel
Zabir Rahman
Systems Programming
Assignment 0: Insertion & Quicksort

Overview: code reads in data from a given file and places it into linked list structure. Then it sorts the data using quicksort or insertion sort based on user input and outputs the data in ascending order.

How Code Operates?
Our code reads in contents from a file which can either be words or integers and places them into a singly linked list. While reading, our code is making sure to disregard specific tokens such as commas, spaces, new lines, and tabs. Using the linked list structure which is filled with the file data, our code will use either insertion sort or quicksort algorithm to sort and output the data in ascending order. By making sure to disregard the specific tokens, our code will be sure to only place valuable, comparable contents into the nodes. For both sorting methods to work, we created an integer and strings comparator methods which took in void* as inputs. For our string and number comparator method, we used two different nodes as inputs and extracted the contents of the nodes for comparison. If the contents of first node were less than contents of second, we returned 0, else we returned 1. We had an int value signal to our methods whether the file contained integer data or string data and based on this signal, we would employ the necessary comparator method in our sorting functions. Our code also reads in user input flag which signals which sorting method to employ. If flag is "-i", insertion sort is employed, and if flag is "-q", quicksort is used. Insertion sort works by sorting and comparing the contents of each node one-by-one. By using a singly linked list structure we have to unlink and relink the nodes based on our comparators to satisfy the insertion sort algorithm. Insertion sort takes longer to sort on large data sets. Our quicksort algorithm works by linking each node of the LinkedList to an array of pointers (hash table of pointers). So, at index 0 of the hash table, it would point to the first node of the linked list. After all indices of the array have pointed to their designated node on the linked list, the next pointers of the linked list nodes are removed, so we can treat the linked list as an array. We then implement the quicksort method by setting the pivot to the first element and recursively solving the left and right sides until the linked list is completely sorted. I then write out the sorted contents and free my malloced memory.

How to use code:
To use this code, user must understand the proper format of the arguments on the terminal and the compile step that comes first.

- First compile the source code: `~/$ gcc filesort.c -o filesort`
- Run the program(v1): `~/$ filesort.c -i somefile.txt`
- Run the program(v2): `~/$ filesort.c -q somefile.txt`

The -i and -q are sorting flags, if you use -i program will do insertion sort on "somefile.txt" otherwise if you use -q program will do quick sort on "somefile.txt".
Note: the order of the -q or -i and somefile.txt cannot be interchanged or else program will print fatal error.