CSE 15
Introduction to Data Structures
Lab Assignment 5

This short project is actually an exercise discussed in class, namely to convert one of our sorting algorithms so that it operates on arrays of strings instead of arrays of integers. The sorting algorithm you will convert is Quicksort, which can be found at

https://classes.soe.ucsc.edu/cse015/Fall19/Examples/Lecture/SortingSearching/Sort.c

Recall that a string in C is a NUL terminated char array. An array variable is a pointer to its 0^{th} element, so a string variable is of type `char*`, and an array of strings is therefore of type `char**`. Thus, to convert the Quicksort algorithm, one must change the array data type from `int*` to `char**`. The only other change necessary concerns the comparison of strings. Recall that function `strcmp()`, found in the `string.h` library, compares two strings `s1` and `s2` as to their alphabetical order.

$$\begin{aligned} \text{strcmp}(s1, s2) < 0 \quad & \text{iff} \quad s1 < s2 \\ \text{strcmp}(s1, s2) == 0 \quad & \text{iff} \quad s1 == s2 \\ \text{strcmp}(s1, s2) > 0 \quad & \text{iff} \quad s1 > s2 \end{aligned}$$

Begin this project by making the necessary changes to functions `swap()`, `Partition()` and `QuickSort()` at the above examples link. Test your version of Quicksort on some string arrays of your own choosing.

Also convert the function `printArray()` from the same example so that it prints out an array of strings to a given file stream, with each string printed on its own line. The headings for your four required functions will therefore be:

```
void printArray(FILE* out, char** A, int n)
void swap(char** A, int i, int j)
int Partition(char** A, int p, int r)
void QuickSort(char** A, int p, int r)
```

To complete the project, write a program called SortStrings.c containing the above functions along with a function `main()` that takes two command line arguments specifying an input file and an output file. The input file will begin with a line containing a single integer n, giving the number of remaining lines in the file. Each subsequent line will contain one word and nothing else. Open the input and output files, read the first line and parse the integer n. Use that value to allocate an array of strings of length n from heap memory. Next, read in the n strings and place them in the string array, allocating sufficient memory for each string (including the NUL terminator) from heap memory. Sort the array using `QuickSort()`, then print it to the output file using your function `printArray()`. Free all heap memory, and close all files before function `main()` returns. An example pair of input-output files is given below.

Input file:
```
6
happy
sad
foo
bar
aware
igloo
```

Output file:
```
aware
bar
foo
happy
igloo
sad
```

Reading and parsing strings from a file, and writing to a file are illustrated in the example `ParseFileIO.c` posted on the webpage. A Makefile which you may alter as you see fit is also included.

Submit the files

SortStrings.c
Makefile
README

to the assignment name lab5 before the due date. This could be the easiest project this quarter (with the exception of lab1), but please do not wait until the last minute.