COP 3502C Programming Assignment#5
Topics covered: Binary Heap-Priority Queue

Please check Webcourses for the Due Date
Read all the pages before starting to write your code

Introduction: For this assignment you have to write a c program that will utilize the merge sort, insertion sort, and binary search algorithm.

What should you submit?
Write all the code in a single main.c file and submit the main.c file and memory leak detector files in the submission system. Also, submit a text file yourlastname.txt explaining the bigO run-time of your code.

Please include the following lines in the beginning of your code to declare that the code was written by you:

```
/* COP 3502C Programming Assignment 5
This program is written by: Your Full Name */
```

Compliance with Rules: UCF Golden rules apply towards this assignment and submission. Assignment rules mentioned in syllabus, are also applied in this submission. The TA and Instructor can call any students for explaining any part of the code in order to better assess your authorship and for further clarification if needed.

Caution!!!
Sharing this assignment description (fully or partly) as well as your code (fully or partly) to anyone/anywhere is a violation of the policy. I may report such incidence to office of student conduct and an investigation can easily trace the student who shared/posted it. Also, getting a part of code from anywhere will be considered as cheating.

Deadline:

See the deadline in Webcourses. If you can submit it by the deadline you will get 5% extra. If you can submit it by the late submission deadline, there will not be any penalty. After that the assignment submission will be locked. An assignment submitted by email will not be graded and such emails will not be replied according to the course policy.

Additional notes: The TAs and course instructor can call any students to explain their code for grading.

Problem: The Median

In general, the Median is the ***"middle"*** of a **sorted** list of numbers. For example, for the following list of sorted number: 10, 11, 13, **15,** 16, 23, 26, the median is 15 as we have a total of 7 numbers and after sorting 15 is the middle number on this list. However, if we have even number of items, we have two numbers in the middle and in that case the median will be the average of those two numbers.

In this assignment, you will receive an **unordered stream** of distinct strings where the strings will contain only single word and lower case letters. The maximum length of a string can be 50. As soon as you receive a string, you need to print the median of the strings you have received **so far**. If you have even number of strings, you should print both of the middle strings in sorted order (based on the following definition) as part of the Median.

In order to compare strings, you will use the following rule:

- Given strings a and b, we say a < b if a has fewer characters than b.
- If two strings a and b have the same length, we say a < b if a comes before b in alphabetical order.

Example:

Let's say the data stream are *orange, apple, app, mango, cherry, peach, melon.*
After reading the first item "orange" -> the median is "orange"
After reading the second item "apple" -> the median is "apple orange" *(according to our definition as we have even number of items and apple < orange)*
After reading the third item "app" -> the median is "apple"
After reading the fourth item "mango" -> the median is "apple mango"
After reading the fifth item "cherry" -> the median is "mango"
After reading the sixth item "peach" -> the median is "mango peach"
After reading the seventh item "melong" -> the median is "melon"

Implementation restriction and important hints:

- Note that you are getting one string at a time and then print the median immediately.
- You must implement a compareTo function that will receive two strings and return a positive number if the first string is greater than second string based on the specified rule. Otherwise, it should return a negative number if the first string < second string.
- If there are a total of n strings, the run-time complexity has to be O(n log n) [this O(n log n) restriction does not include how you will take the input.]
- Using insertion sorting technique during this process could be a good idea, however, it will result in O(n^2). So, you are **not allowed** to use any sorting algorithm in your code.
- You must use priority queue (heap) to solve this problem. The following hints should give you an idea of using priority queue in this context.
- **Important hints:** You might be now thinking that how do we really use heap in this problem and how can we find median without sorting! Here is the trick:
    - You can have two heaps, one will store smaller half of the numbers (lefth) and the other one (righth) will store bigger half of the data. One of them will be min heap and the other will will be maxheap. Please think a bit which one will be minheap and which one will be max heap.

- First string is a special case and it is a median without any comparison. Print this, and add it to the heap for the smaller number. Keep track this as a current median.
- **Create three cases:** one for if the size of left heap is greater, another for if the size of the right heap is greater, and finally one for if both heaps have same size.
- As soon as you receive a string, depending on the size of the left and right heap, and the new string and the current median, you might need to delete an item from on heap and insert it to another heap (kind of transferring) during this process to balance the heaps. Also, you need to insert the new string to an appropriate heap.
- Now, you can print the new median from the appropriate heap and update the current median. In case of even items, you can update the current median with the smallest one among them for further processing.
- Consider adding a peek function for lookup the root of a heap. This can be useful in many cases.

- You have to use the heap code we have seen in the class and need to modify it to solve this problem. A good idea would be adding another parameter to the percolateUp and percolateDown function to decide whether the operation is for a minheap or maxheap. Also, remove all unnecessary functions that are not relevant to this. **Any irrelevant unused function in the code may result in penalty.**
- Your code should contain at least the following heap related functions: Insert, Percolateup, Removemin, Removemax, initheap, swap, minimum, maximum.
- During this process, string operations could create additional challenges. Make sure to do proper dynamic memory allocatin, strcpy, etc.
- You must use the memory leak detector and In order to get full credit, you need to free all the memory.
- The code will be tested in **codegrade** platform like previous assignments.
- The output must have to match with the sample output format. Do not add additional space, extra characters or words with the output as we will use diff command to check whether your result matches with our result.
- Next, you must have to write a well structure code. ***There will be a deduction of 10% points if the code is not well indented and 5% for not commenting important blocks.***

The Input (to be read from standard console intput) - Your Program Will Be Later Tested on Multiple inputs

The first line of the input contains 1 integer n (0 < n < 10,001) that indiecates the number of strings. Then the ne lines will contain n distinct strings where each string is a single word with all lower case letter. The maximum length of a string will be 50.

The Output (to be printed to console as well as to out.txt file)

There should be n lines of output. The ith line line of the output should contain the median at that moment based on the ith input.

Sample Input

```
7
orange
apple
app
mango
cherry
peach
melon
```

Sample Output (out.txt as well as in standard console output)

```
orange
apple orange
apple
apple mango
mango
mango peach
melon
```

Steps to check your output AUTOMATICALLY in shell:

You can run the following commands to check whether your output is exactly matching with the sample output or not.

Step1: Copy the sample output into sample_out.txt file and move it to the server (you can make your own sample_out.txt file)

Step2: compile and run your code using typical gcc and other commands. Your code should produce out.txt file.

```
$gcc main.c leak_detector_c.c
$./a.out <in.txt
```

Step3: Run the following command to compare your out.txt file with the sample output file

```
$diff -i out.txt sample_out.txt
```

The command will not produce any output if the files contain exactly same data. Otherwise, it will tell you the lines with mismatches.

Incase if your code does not match, you can use the following command to see the result in side by side:

```
$diff -y out.txt sample_out.txt
```

Rubric (Suject to change):

- A code not compiling or creating seg fault without producing any result can get zero. There may or may not be any partial credit. But at least they will not get more than 50% even if it is a small reason. Because, we cannot test those code at all against our test cases to see whether it produces the correct result or not.
- Missing any **one** of the restriction can result in 50% penalty. More than one may result in 0 in the assignment.

- There is no grade for just writing the required functions. However, there will be 20% penalty for not writing and using CompareTo() function
- Reading the input properly: 4 pts
- Run-time analysis: 8 pts
- Properly freeing memory leak: 6 pts
- Implementing both min heap and max heap properly with proper use of strings and compareTo function for this assignment scenario: 32 pts
    - Insert, Percolateup, Removemin, Removemax, initheap, swap, minimum, maximum.
- Generating proper output and passing all test cases (note that more test cases can be added while grading. So, you should test your code properly and consider generating your own test cases): 50
    - There will be various types of simple to complex test files within the assignment criteria
- There is no point for well structured, commented and well indented code. ***There will be a deduction of 10% points if the code is not well indented and 5% for not commenting important blocks.***