COSC2430: Data Structures and Programing
Sorting Algorithms

1 Introduction

You will create a C++ program that can use five sorting algorithms (select, insert, quick, merge, heap) to sort singed infinite integers and analysis the bigO of the sorting algorithms (the number of comparisons and the number of swaps) . Each singed infinite integer should be stored in one DoubleLinkedList. You can store all singed infinite integer in an array of DoubleLinkedList.

2 Input and Output

The input is a regular text file, where each line is terminated with an end-of-line character(s). Each line will contain a singed infinite integer. The program should output the results to a regular text file. Each run should append the analysis to bigO.txt file.

Input example: input.txt

```
1
-1
-2
0
-2000000000000000
2
1
10000000000000000
999999999999999999
12345667890123456789
8765432109876543210
```

Output example: output.txt

```
-2000000000000000
-2
-1
0
1
1
2
10000000000000000
8765432109876543210
12345667890123456789
999999999999999999
```

3 Program input and output specification

The main program should be called ”sort”. The output should be written to the a regular text file. Only generate one bigO.txt.

Call syntax at the OS prompt (notice double quotes):

```
sort "input=<in.txt>;digitsPerNode=<number>;algorithm=<select|insert|merge|heap|quick>;ouput=<out.txt>".
```

Example of bigO.txt file:

```
algorithm  #numbers(n)  #Comparisons #Swaps #bigOComparisons #bigOSwaps
heap         10            40           8        45             50
quick        100          2000          77      5050            5050
select       10            45           10      45              45
insert       100          5050          0       5050             0
...
```

Where #bigOComparisons and #bigOSwaps are the worst case given by the input n numbers. bigO.txt is manually graded by TAs. bigO.txt counts 20 credits of this homework.

Assumptions:

- The file is a small plain text file (say < 10000 lines); There are up to 256 characters in each line; no need to handle binary files.
- Only integer numbers as input (no decimals!). Output number without leading zeroes and character ’+’.

Example of program call:

```
sort "input=in1.txt;digitsPerNode=5;algorithm=heap;ouput=out1.txt"
```

4 Requirements

- Correctness is the most important requirement: TEST your program with many input files. Your program should not crash or produce exceptions.
- Doubly linked lists are required. A program using arrays(std::vector, std::string) to store long numbers will receive a failing grade (below 50).
- Breaking a number into a list of nodes. Each node will store the number of digits specified in the parameters. Notice it is acceptable to ”align” digits after reading the entire number so that the rightmost node (end) has all the digits.
- input and output numbers. The input and output numbers must be stored on lists.
- Memory allocation and deallocation.

  Your program must use ”new” to allocated each node and must free memory (using the ”delete” operator). at the end.
- Limits: Each node will store a fixed number of digits, specified with the ”digits per node” parameter. You can assume 1-8 digits per node.