IC250 Lab 5

This assignment has three compulsory questions and one optional question.

1. **Merging.** Merge two sorted arrays into a single input array. The two arrays must be read from two input files. A sample run is shown below:

```
$ cat input1 <ENTER>
3
4
7
8
$ cat input2 <ENTER>
4
7
9
12
15
16
$ ./merge input1 input2 <ENTER>
Reading file input1...
Reading file input2...
The merged output is
3
4
4
7
7
8
9
12
15
16
$
```

2. **Mergesort on array of records.** A file of employee records has the following structure:

```
struct node {
   int index; // record-key
   char[20] fname, lname, dob, location, dept;
}
```

Read an input file of records into an array. Sort the records on the `index` field using mergesort. Remember, this is an *array* of records. Write out the sorted records into a new file. Note that the file could have several hundred records, and this can be provided to your program as an input. If no command line arguments are specified, print the usage. A sample run is shown below.

```
$ cat data <ENTER>
 28053   Rahul Ranjan 28-Nov-2001 Kolkata CustomerCare
 31163   Shahid Gupta 15-Aug-1988 Delhi CustomerCare
  3786   Meena Srivas 05-Jan-1990 Lucknow Spares
 33086   Santosh Singh 05-Jan-1990 Chennai SupplyChain
 16355   Meena Singh 23-June-1999 Kolkata Sales
  4792   Santosh Srivas 28-Nov-2001 Delhi CustomerCare
 26524   Meena Nath 23-June-1999 Panjim Production
 21076   Meena Andrews 12-Feb-1976 Kolkata Service
 21367   Ben Gupta 05-Jan-1990 Chennai Spares
 27286   Meena Andrews 10-Oct-1980 Kolkata Marketing
$ ./mergesort <ENTER>
Usage: mergesort infile outfile numberOfRecords
$ ./mergesort data sortedData 10 <ENTER>
Reading file data...
Performing mergesort on array...
Output written into file sortedData...
Done.
$ cat sortedData <ENTER>
  3786   Meena Srivas 05-Jan-1990 Lucknow Spares
  4792   Santosh Srivas 28-Nov-2001 Delhi CustomerCare
 16355   Meena Singh 23-June-1999 Kolkata Sales
 21076   Meena Andrews 12-Feb-1976 Kolkata Service
 21367   Ben Gupta 05-Jan-1990 Chennai Spares
 26524   Meena Nath 23-June-1999 Panjim Production
 27286   Meena Andrews 10-Oct-1980 Kolkata Marketing
 28053   Rahul Ranjan 28-Nov-2001 Kolkata CustomerCare
 31163   Shahid Gupta 15-Aug-1988 Delhi CustomerCare
 33086   Santosh Singh 05-Jan-1990 Chennai SupplyChain
```

---

3. **Mergesort on a linked list of records.** Using the same record structure as above, read the input file of records into a linked list. The number of elements is *not* provided to the program, and there could be potentially thousands of records (or even more!). This time, sort the linked list of records using mergesort.

   Hint: To use mergesort, you need to keep dividing the list into two halves. This can be done by a linear traverse with two pointers, one going two steps ahead of the other. At the end of the traverse, the slower pointer will be at the middle and the faster one at the end.

   The sample run will be similar to that of the previous question, except that you dont provide the size of the input to the program. A sample file of records is provided on Moodle to check your program.

4. **Optional:** Generate a large sequence of integers and shuffle it. Store the shuffled sequence into an array and into a linked list. Run mergesort on both. Which data structure is better? In other words, is sorting an array using mergesort faster than sorting a likned list?

   Note. You must use a really large sequence ($> 10,00,000$) to check the timing. Use functions from time.h to check the timing.