**Homework 15 – Course Project – External Merge Sort  
Due: Sunday (see Syllabus)  
Points: 75  
  
Instructions**In this assignment you will write a C program (**ExternalSort.c**) that sorts large files containing many random integers.   
  
The main program in **ExternalSort.c** program should call a function:   
**void externalSort(char\* inputFileName, char\* outputFileName, int bufferSize)**, where  
  
 1. **inputFileName** – the name of the input file containing the random ints.  
 2. **outputFileName** – the name of the output file containing the sorted ints.  
 3. **bufferSize** –the size of the internal buffer used by the internal sort function.  
  
If the program cannot read the input file or cannot create file **outputFileName**, an appropriate message should be displayed, and the program should terminate.  
  
If **bufferSize** is outside the range 0-200,000, an appropriate message should be displayed, and the program should terminate.  
  
Your **externalSort()** function must use an **external merge sort algorithm** to sort the data in the input file into numeric ascending order using a collection of temporary files to hold the intermediate results, writing the sorted ints into the output file. Pseudocode for performing an external merge sort is given below.  
  
Your **externalSort()** function should display the number of ints found in the input file as well as the number of temporary files that were used during the sorting process. These temporary files should be deleted by your program when it terminates.  
  
Your **externalSort()** function must time its activity reporting how much time it took to create the random number output file.   
  
So, you should use the random number generator provided in **Random.c** (attached) rather than the built-in **rand()** function.  
  
See the Sample Output below for what messages your program should display and in what form they should be presented. Note that large numbers should be displayed with commas to set off thousands. For example, 12,345,678.  
  
Run your program several times using different inputs – sufficient to demonstrate that your program meets all the assignment requirements. . For your graded submission, your main must call **externalSort ()** 5 times using the following inputs:  
  
 **Input File** **Output File** **Buffer Size**  
 data5K.txt sorted5K.txt 100  
 data40K.txt sorted40K.txt 1000  
 data300K.txt sorted300K.txt 10000  
 data2M.txt sorted2M.txt 100000  
 data10M.txt sorted10M.txt 200000  
  
Capture screen shots of your run and paste them into an MS Word document. Place a caption above each image.  
  
**Submit MS Word documents: one each for your C program source files (the .c files and .h files) and one for your screen shot(s). Submit using the appropriate Assignment Submissions link.  
  
  
Attach: Hw13-CustomRandomAndElapsedTime.zip** Contains **Random.c** and **ElapsedTime.c  
  
  
  
External (Merge) Sort Algorithm**Sometimes the amount of data to be sorted (stored in a disk file) is too large to fit into available memory. In these situations, an “external” sort must be used to create an output file that contains all of the original data in sorted order. A variation of the traditional merge sort algorithm can be used to perform the sort using temporary files to contain the intermediate results. The size of the intermediate results must be small enough to fit into available memory.  
  
**Pseudocode for External Merge Sort (this is a “k-way sort”)**

1. Obtain the names of the input file and output file and the size of the memory buffer to be used. Call this **length**. Temporary files will contain at most **length** values.
2. Record the current time.
3. Allocate an int buffer of size **length**.
4. *Distribute:* Repeatedly:
   1. Read **length** values from the input file storing them in the buffer.
   2. Sort the buffer.
   3. Write a new temporary file containing all these values. You will need to keep track of these temporary file names so that they can be used in step 6.
   4. Repeat until all input values have been read, sorted, and output.
5. You now have **k** temporary files.
6. *Merge:* Read the first value from each of the **k** temporary files saving them in an array of length **k**.

Repeatedly:

* 1. Find the smallest value in the array and output that value to the output file. Each time you do so, read another value from the corresponding temporary file to replace the value just output.
  2. Continue to do this until all temporary files have been completely read and all the values in the array have been sent to the output file.

1. Calculate the elapsed time.
2. Display the final results.

**NOTES**

* The last temporary file may contain fewer than **length** values.
* You should use the built-in C **qsort()** function to sort the buffer in step 4.  
  See <https://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm> for   
  details on how to use **qsort().**
* Your program should clean up (delete all temporary files) before it terminates.
* Sanity check: the file sizes of the input file and the output file should be identical. You should check this.
* If your program takes more than 10 minutes to sort any of the data files you should carefully examine your code. The Sample Output shows that sorting 10 million ints took just over 60 seconds on a mid-range personal computer.

The Sample Output is on the next page

**Sample Output from ExternalSort**A screenshot of a computer program

Description automatically generated