**Homework 14 + 15**

**Due: Sundays (see Syllabus)**

**Part 1 Points: 20**

**Part 2 Points: 60**

**Instructions**

This is part 2 of a 2-part assignment to create an external merge sort program in Perl. The details are:

* Part 1 – Write a Perl program (***CreateBigData.pl***) that creates a file containing a specified number of random integers.
* Part 2 – Write a Perl program (***ExternalSort.pl***) that sorts a file containing random integers.

**NOTE**

You will need to complete Part 1 before Part 2 because you’ll need to use **CreateBigData.pl** to create the data files needed by **ExternalSort.pl**.

**PART 2 - DETAILS: *ExternalSort*.pl**

The **ExternalSort.pl** program should be a “console program” which is run from the command line. The usage is:

**perl ExternalSort.pl inputFileName outputFileName bufferSize**,

where:

**inputFileName** is the name of the data file to be sorted.

**outputFileName** is the name of the file to be created containing the sorted data.

**bufferSize** is the size of the internal buffer used by the internal sort procedure.

If **ExternalSort.pl** is run without 3 arguments, a Usage message should be displayed, and the program should terminate.

If the program cannot read the input file or cannot create file **fileName**, an appropriate message should be displayed, and the program should terminate.

If **bufferSize** is outside the range 0-100,000, an appropriate message should be displayed, and the program should terminate.

Your program should 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 program 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.

Your program must time its activity reporting how much time it took to create the random number output file. See **ElapsedTime.pl** (from Exercise 14) for an example of how to time your program’s calculations.

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, you should use the **RunEx15.bat shell** script file to create your submission data files. Note that you will need the 5 data files created by Part 1 (data5K.txt, data40K.txt, data300K.txt, data2M.txt, data10M.txt). Capture a screen shot of each run and paste them into an MS Word document. Place a caption above each image.

**Submit the Perl .pl file (lastname\_hw14.pl) containing your program and the MS Word document** **to your instructor using the appropriate Assignment Submissions link.**

Attached: ***ElapsedTime.pl***

***RunEx15.bat***

**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 may use the built-in Perl **sort** function to sort the buffer in step 4.
* 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.

**Sample Output (ExternalSort.pl with a buffer size of 50):**

Text

Description automatically generated

**Sample Output (RunEx15.bat):**

Running ExternalSort 5 Times...

External Sort

Sorting file: data5K.txt

Output file: sorted5K.txt

Buffer size: 100

File data5K.txt sorted. 5,000 integers. 50 temp files.

Elapsed time:

0.002 minutes

0.102 seconds

102.1 milliseconds

Done

External Sort

Sorting file: data40K.txt

Output file: sorted40K.txt

Buffer size: 1,000

File data40K.txt sorted. 40,000 integers. 40 temp files.

Elapsed time:

0.007 minutes

0.394 seconds

393.9 milliseconds

Done

External Sort

Sorting file: data300K.txt

Output file: sorted300K.txt

Buffer size: 10,000

File data300K.txt sorted. 300,000 integers. 30 temp files.

Elapsed time:

0.027 minutes

1.621 seconds

1621.2 milliseconds

Done

External Sort

Sorting file: data2M.txt

Output file: sorted2M.txt

Buffer size: 100,000

File data2M.txt sorted. 2,000,000 integers. 20 temp files.

Elapsed time:

0.140 minutes

8.402 seconds

8401.8 milliseconds

Done

External Sort

Sorting file: data10M.txt

Output file: sorted10M.txt

Buffer size: 200,000

File data10M.txt sorted. 10,000,000 integers. 50 temp files.

Elapsed time:

0.766 minutes

45.964 seconds

45963.6 milliseconds

Done

5 Runs Completed