CURTIN UNIVERSITY (CRICOS number: 00301J)

Department of Computing, Faculty of Engineering and Science

Data Structures and Algorithms (COMP1002)

PRACTICAL 1 – FILES AND SORTING

AIMS

- To implement and test Bubble Sort, Insertion Sort and Selection Sort
- To parse and sort the data in a file.

BEFORE THE PRACTICAL:

- Read this practical sheet fully before starting.
- Copy the class Sorts.java or Sorts.py from Blackboard. Use this as a starting point to writing the sorting methods.
- Download the SortsTestHarness to test your code.
- Write code to use the data from RandomNames7000.csv to test ALL of your sorting algorithms.

ACTIVITY 1: BUBBLE SORT IMPLEMENTATION

Time to write some code:

- In Sorts.java, implement the bubbleSort() method using the pseudocode from the lecture slides as a guide.
- Note that the method in the Sorts starter file works on an int[] array.
- Don't forget to include a check to stop bubble sort if it doesn't do any swaps during a pass (*i.e.,* the array has finished being sorted).
- Test your code using the SortsTestHarness.java/.py.

ACTIVITY 2: SELECTION SORT IMPLEMENTATION

Do the same for Selection Sort as you did for Bubble sort:

- Selection sort differs from Bubble sort in that it only swaps *once* per pass. Instead, what it does is searches for the smallest value, updating the *index* of the smallest value until the end of the pass. Only then does it swap the smallest value with the first value.
  - Remember that in the second pass, the first value has already been sorted. So don't include the first value in the second pass.
  - The same is true for subsequent passes (ie: the third pass should ignore the first two values).
- Test your code using the SortsTestHarness.java/.py.

ACTIVITY 3: INSERTION SORT IMPLEMENTATION

Do the same for Insertion Sort as you did for Bubble and Selection sort

ACTIVITY 4: EXPLORING RUN TIMES

The SortsTestHarness lets you easily test your sorts code for various types of data

- Develop a table of runtime results, using at least four data sizes and two random/sorted options across the three sorts
- Write a paragraph discussing the results in terms of complexity and other characteristics of the three sorts

ACTIVITY 5: SORTING A FILE

Now you can try sorting a file of random student numbers and names:

- Parse the file to read in the student ID's
- Sort the data using each of the sorts
- Output the sorted list to a file
- You can read the ID and name into an object as a challenge (not required)

SUBMISSION DELIVERABLE:

Your code, data and document (For Activity 4) are due before the beginning of your next tutorial. Also include any other relevant classes that you may have created. Java students, please do not submit the *.class files.

You must submit your code and any test data you've been using.

SUBMIT ELECTRONICALLY VIA BLACKBOARD, under the *Assessments* section.

MARKING GUIDE

Your submission will be marked as follows:

- [2] x3 Each Sort is implemented properly and tests correctly.
- [2] sorts performance investigation.
- [2] file reading and sorting.