Curtin University

School of Electrical Engineering, Computing & Mathematical Sciences

PRACTICE FINAL ASSESSMENT

End of Semester 1, 2020

COMP1002 Data Structures and Algorithms

This paper is for Curtin Bentley, Mauritius and Miri students

This is an OPEN BOOK examination

Examination paper IS to be released to student

Examination Duration	24 hours	**Start** 3:00pm 1st June, **Finish** 3pm 2nd June
Reading Time	N/A	

- Unit Coordinator will be available Collaborate for first four hours to respond to questions

Total Marks 10

Supplied by the University

- Exam paper
- Source material for Q1-2 in zip file
- Collaborate and email access to Tutors/Unit Coordinator (v.maxville@curtin.edu.au)

Supplied by the Student

- Linux command line, all java code must be compilable using javac *.java

Instructions to Students

- Attempt all questions.
- Open book/computer, however, you must cite references.
- Code for each task **must run** to be awarded any marks.
- Keep all work within a directory: PracExam_<Student_ID>.
- Copied code will receive zero (0) marks
- Students must not work together or get help from other people
- **When complete:**
  - Generate a history file (history > hist.txt)
  - Sign Declaration of Originality
  - Create a zip file of the Prac Exam directory (-r for recursive zip)
  - Submit to Assessment link on Blackboard before 3:00pm 2nd June (Perth-time)
  - IF there are problems uploading to Blackboard, submit to COMP1002@curtin.edu.au
- **All submissions will be subjected to rigorous testing for plagiarism, collusion and any other forms of cheating. You must cite any and all design/java/python from any source, including your own work submitted for a different assessment.**
- **Assessment may include a follow-up demonstration or interview (viva)**

QUESTION ONE (Total: 5 marks): Sorting

a) Access your practicals to get a copy of Sorts.java/Sorts.py. We need the following sorts. Implement them if you do not already have them:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Two versions of Quicksort
    - *i.* Pivot strategy 1 is Rightmost
    - *ii.* Pivot strategy 2 is Median of 3

**** Don't forget to reference/cite previously submitted code*

b) You need to write a test harness (PFASortsHarness.java/py) to read in files and output the timing. You might use the SortsTestHarness as a starting point.

**** Don't forget to reference/cite sources*

c) Edit the **text** file (PFASortsAnalysis.txt) to discuss performance for the listed Sort/File combinations. This is what we will assess – one mark per sort. Discussion should be a few paragraphs per sort, relating the performance to the theoretical performance for each sort.

PFASortsAnalysis.txt

Bubble Sort

File1=xxx File2=xxx File3=xxx File4=xxx

Discussion :

Insertion Sort

File1=xxx File2=xxx File3=xxx File4=xxx

Discussion :

Selection Sort

File1=xxx File2=xxx File3=xxx File4=xxx

Discussion :

Quicksort (Rightmost)

File1=xxx File2=xxx File3=xxx File4=xxx

Discussion :

Quicksort (Median of 3)

File1=xxx File2=xxx File3=xxx File4=xxx

Discussion :

QUESTION TWO (Total: 5 marks): Recursion

a) We will be using the following algorithms to give a performance profile and discussion of each. Implement them if you do not already have them:

- Iterative Factorial
- Recursive Factorial
- Iterative Fibonacci
- Recursive Fibonacci
- Recursive Fibonacci (with solution caching)

**** Don't forget to reference/cite previously submitted code*

b) You need to write a test harness (PFARecursionHarness.java/py) to call the various algorithms and output the timing. Again, you might use the SortsTestHarness as an inspiration.

**** Don't forget to reference/cite sources*

c) Edit the **text** file (PFARecursionAnalysis.txt) to discuss performance for the recursive and iterative algorithms. This is what we will assess – one mark per algorithm. Discussion should be a few paragraphs per algorithm, explaining your reasoning for the different performance of each.

PFARecursionAnalysis.txt

Iterative Factorial

Input data + performance

Discussion

Recursive Factorial

Input data + performance

Discussion

Iterative Fibonacci

Input data + performance

Discussion

Recursive Fibonacci

Input data + performance

Discussion

Recursive Fibonacci (with solution caching)

Input data + performance

Discussion

END OF EXAMINATION