

Note: This is an individual assignment. While it is expected that students will discuss their ideas with one another, students need to be aware of their responsibilities in ensuring that they do not deliberately or inadvertently plagiarize the work of others.

Description

This assignment consist of 3 main tasks:

- a) Task 1 - Implementation and performance analysis of computation algorithms [8 marks]
- b) Task 2 - Implementation and performance analysis of sorting algorithms [8 marks]
- c) Task 3 - (Advanced) Research, conclusions of statistical analysis and suggestions for improvement [4 marks]

Note : For all the main parts (mentioned above), you are free to choose the development platform IDE (e.g. Visual Studio, Netbeans, etc) and programming language (e.g. Python, C++, Java, etc) you are familiar with, in order to fulfill all the necessary task requirements.

Task 1

For the first part of this assignment, you are required to develop 2 functions :

- a) A **random-number-generator** function, that generates and returns a series of random numbers
- b) A **save-to-file** function to save a set of random numbers into a simple text (*.txt) file (which could later be opened by simple text editor like Microsoft Notepad / Wordpad)

To aid you in your design, the tables (below) illustrates the details of each function, its inputs / outputs, and provides a basic description of its purpose.

James Cook University
CP5602 Assignment (20%)
 Due: at Lab 9

	Random Number Generator function
Suggested function name / signature	<code>long [] rNGenerator (long lowerLimit, long upperLimit, long numOfValues, boolean noDuplicates)</code>
Function Output	<code>long []</code> An array of whole numbers with data type 'long'. You are free to choose any 'whole' number datatype available in your programming language, as long as it can store a very large range of values (e.g. in Java, there is a "BigInteger" datatype which you may consider as well)
Function Inputs (arguments)	<code>lowerLimit</code> The lowest (whole number) value your function can generate (E.g. if <code>lowerLimit = -3</code> , you cannot generate a random value lower than -3 !!)
	<code>upperLimit</code> The highest (whole number) value your function can generate (E.g. if <code>upperLimit = 8000</code> , you cannot generate a random value higher than 8000 !!)
	<code>numOfValues</code> The total number of random values your function should generate. (E.g. if <code>numOfValues = 18000</code> , your function should generate and return an array of 18000 random values within the <code>lowerLimit</code> to <code>upperLimit</code> range!)
	<code>noDuplicates</code> This is a Boolean value (i.e. T/F) If <code>noDuplicates = true</code> , each value generated by your function should be unique (i.e. no duplicates)! If <code>noDuplicates = false</code> , it is possible some of the random values generated by your function can be repeated ...

	Saving to file function
Suggested function name / signature	<code>void saveToFile (long [] generatedRandomNumber, String filename)</code>
Function Output	<code>void</code> The main purpose of this function is to save an array of random number data (generated by the random generator function), into an external text file. As such, the function itself has no return type (hence 'void')

James Cook University
CP5602 Assignment (20%)
 Due: at Lab 9

Function Inputs (arguments)	generatedRandomNumber The random number data to be stored. The suggested datatype for this input is an array of <code>long</code> datatype, however, you can decide the data type for this array as long as it can store a large range of whole number values
	filename The filename to save all the generated random number data

Task 1 - Requirements

- a) Design various test scenarios, to test your `random-number-generator` function with different sizes of input (`n`). For example:
 - Test **A1** – generate `n` = 50000 random values from -35500 to 36600
 - . . . (you can design many other intermediate tests with different '`n`' input sizes or lower to upper limit ranges) . . .
 - Test **An** – generate `n` = 80000 random values from -13800 to 96800
 - Test **B1** - generate `n` = 50000 **unique**, random values from -35500 to 36600
 - . . . (you can design many other intermediate tests with different '`n`' input sizes or lower to upper limit ranges) . . .
 - Test **Bn** - generate `n` = 80000 **unique**, random values from -13800 to 96800
- b) Write a "task 1" program to display (for each test), the time taken to:
 - generate the required number of random values **AND**
 - store all of them to an external text file (using `save-to-file` function)
- c) Plot **TWO** graphs based on the information gathered above (for consistency, all input size (`n`) values should be placed on the x-axis, and algo's timing performance should be represented on y-axis)
 - Graph #1 – the timing results for Test **A1** – **An** (non-unique random values)
 - Graph #2 – the timing results for Test **B1** – **Bn** (unique random values)
- d) For each graph in step c), determine and write down the Big-O characterization, in terms of `n`, on the performance of your `random-number-generator` & `save-to-file` functions

Task 2

For this task, you are to implement 2 sorting algorithms: insertion sort and quick sort which will take in a array of values, and proceed to sort them in ASCENDING order. The algorithm / pseudo code for both sorting algorithms can be found in the following:

- Module 2 (Chapter3.pdf), slide 9 - for insertion sort
- Module 7, (Chapter11.pdf), slides 9-10 - for quick sort

Task 2 - Requirements

- a) The sorting functions should be implemented using the same programming language as Task 1.
- b) Please make use of your random-number-generator function (implemented for Task 1) to generate random whole number values **as inputs** to each of your sorting function (i.e. insertion and quick sort)
- c) Please make use of your save-to-file function (implemented for Task 1) to store the sorting results (i.e. sorted numbers) for each of your sorting function
- d) Design various test scenarios, to test each sort function with different sizes of input (**n**).

For example:

- Test **C1** – generate **n** = 50000 random values from -35500 to 36600, pass these random values into your insertion sort for processing, then save to a text (*.txt) file
- . . . (you can design many other intermediate tests with different '**n**' input sizes or lower to upper limit ranges) . . .
- Test **Cn** – generate **n** = 80000 random values from -13800 to 96800, pass these random values into your insertion sort for processing, then save to a text (*.txt) file
- Test **D1** - generate **n** = 50000 random values from -35500 to 36600, pass these random values into your quick sort for processing, then save to a text (*.txt) file
- . . . (you can design many other intermediate tests with different '**n**' input sizes or lower to upper limit ranges) . . .
- Test **Dn** - generate **n** = 80000 random values from -13800 to 96800, pass these random values into your quick sort for processing, then save to a text (*.txt) file

James Cook University
CP5602 Assignment (20%)
Due: at Lab 9

- e) Write a "task 2" program to display (for each test), the time taken to:
- sort the required number of random values **AND**
 - store all of them to an external text file
- f) Plot **TWO** graphs based on the information gathered above (for consistency, all input size (n) values should be placed on the x-axis, and algo's timing performance should be represented on y-axis)
- Graph #1 – the timing results for Test **C1** – **Cn** (insertion sort algo)
 - Graph #2 – the timing results for Test **D1** – **Dn** (quick sort algo)
- g) For each graph in step f), determine and write down the Big-O characterization, in terms of n, on the performance of your sorting function algorithm

You are to gather sufficient data points, to plot an accurate graph that will help you to determine the Big-O characterization, in terms of n, of the Best, Average and Worst processing time for both sorting algorithms (express your answer in a table format as shown below)

Sort Algorithm	Sorting Time		
	Worst	Average	Best
Insertion Sort			e.g. $O(n \log(n))$
Quick Sort			

Task 3 (Advanced)

The tasks for this part depends on the successful and correct implementation of Parts 1 - 3. You are to use your own initiative and creativity to design performance tests that will answer the following questions:

- a) Does the no. of CPUs in a machine significantly affect the performance of the unknown algorithms (in Part 1), sorting algorithms (in Part 2) and data structures (in Part 3)?
- b) Does having greater RAM (Random Access Memory) capacity help improve performance for all input sizes?
- c) Does the kind of Operating System running your programs play an important role, in maximizing your performance in Parts 1-3? (To answer this, you must choose at least 1 O.S. from the Windows platform, and at least 1 O.S. from the Unix platform)

You are to justify all your answers based on the actual results gathered in your performance.

The quality of your answers depends on the variety, scope and depth of your performance test coverage.

In situations where it is not possible to present a consistent or general conclusion, you should **qualify your answers** by describing all the different scenarios where your results do not follow certain trends established by your results. E.g. *"... for input of certain sizes (insert details here), the kind of O.S. does not affect the performance significantly. But beyond this size, for such-and-such algorithm, in such-and-such circumstances, O.S. "A" with 2 CPUs and 16Gb RAM does better than O.S. "B" ..."*

Deliverables

(I) What to include

- 2 programs (Task1 & Task2), each compiled into Windows executable (*.exe) - please make sure it can run on your lab PC's Windows OS!
- The text file (*.txt) output, for each task's program, based on your designed test scenarios
- MS Word Report, with statistics, screen-shots of results and conclusions for Parts 1-3
- Excel file with the necessary statistics and graph
- **Note** : A presentation / demo session will be arranged for students to showcase their programs and findings to the tutor and fellow classmates

(II) Naming Conventions

For **all files**, use the following naming format :

<Stud. No.> _ <Name> _ Assn. <file extension>

- <Stud. No.> refers to your JCU assigned student number (e.g. 1234567)
- <Name> refers to your JCU registered name (e.g. JohnDoeAnderson)
- <file extension> refers to the type of file (e.g. .doc, .xls, etc)

Example 1 :

if your name is 'John Doe Anderson' and you are submitting a **Word document**, then the filename would be :

JohnDoeAnderson_1234567_Assn.doc

Example 2 :

if your name is 'John Doe Anderson' and you are submitting a **Excel document**, then the filename would be :

JohnDoeAnderson_1234567_Assn.xls

(III) How to package

Github link will be created to receive the file links. A zip file should also be attached in LearnJCU

(IV) Where to submit

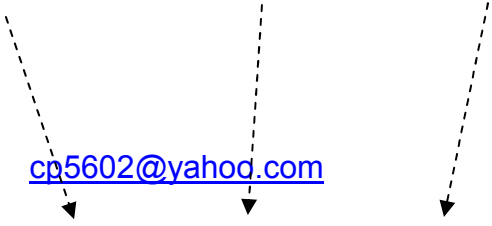
Please email your single zip file to your tutor at : cp5602jcu@yahoo.com

In your email **subject** line, type in the following information :
<assignment information> <student number> and <name>.

Example:

To : cp5602@yahoo.com

Subject : Assn 1234567 JohnDoeAnderson



Note : The timestamp shown on tutor's email Inbox will be used to determine if the assignment is late or not.

(V) When to submit (DEADLINE !)

Please submit at least 1 day before the start of lab 9. On the actual lab, your tutor may randomly pick a task and ask you demo / share your findings with the class!

(VI) ! VERY IMPORTANT !

PLEASE FOLLOW THE GUIDELINES IN ALL APPENDICES !!
PLEASE FOLLOW THE SUBMISSION INSTRUCTIONS FROM (I) TO (V) !!
IF YOU ARE NOT SURE, PLEASE CHECK WITH YOUR TUTOR ...

GITHUB LINK IS MANDATORY

MARKS WILL BE DEDUCTED IF YOU FAIL TO FOLLOW INSTRUCTIONS !!

Example report document or zip file does not follow naming convention

- The structure of your answer is partially / totally different, from those stated in the paper
- Wrong naming or information given
(e.g. putting "Assn" in email subject, when you are submitting "Minor Research Report")
(e.g. naming "Assn" in your zip file, but inside contains other files)
- Your submission cannot be downloaded and/or unzipped
- Your report cannot be opened by Microsoft Word / Excel
- Etc.