GRIFFITH COLLEGE DUBLIN

COMPUTING ASSIGNMENT TITLE SHEET

Course:	BSCH Computing
Stage/Year:	4
Module:	Distributed Systems
Semester:	Semester II
Assignment Number:	Assignment 1
Date of Title Issue:	Nov 11th, 2020
Assignment Deadline:	Dec 3rd, 2020 - 11:55pm
Assignment Submission:	Moodle Upload
Assignment Weighting:	20/40

Standard penalties will be applied to work that is submitted late, as per faculty guidelines.

*All work **must be your own***
If you copy from someone else, both parties will be awarded a grade of 0.

Learning Outcomes

Programme and related module learning outcomes that this assignment is assessing:

1,3

Assessment Criteria

Assessment criteria applied to this assignment, such as:

- ❑ Presentation
- ❑ Code Structure and Cleanliness
- ❑ Code Performance
- ❑ Appropriate Output
- ❑ Written Report

Assignment 1: Ring and Things

Practical Assignment that introduces a basic distributed system application, covering the setup of processes and the basic mechanisms of communication. Learners are expected to synchronise and coordinate distributed processes.

You are required to create **two different mpi programs**, and **one report.**

Name your files: *FIRSTNAME_LASTNAME_STUDENTNUMBER_PART1.cpp*, *FIRSTNAME_LASTNAME_STUDENTNUMBER_PART2.cpp*, and *FIRSTNAME_LASTNAME_STUDENTNUMBER_REPORT.pdf*

Zip your submission for Moodle. Feel free to include any make files. Name your archive **NAME_NUMBER.zip**. The report must be in PDF format.

Ensure your code is well commented, as well as neat and readable. Code that fails to compile will incur a **penalty of 30%.** Work that is not submitted using the correct format will incur a **penalty of 10%.**

*Work that is **submitted late** will incur standard penalties as per faculty guidelines.*

Print your name and student number once to the console for all programs you write.

Part 1 (50%)

You are tasked with creating a sequential ring of 3 nodes.

The initial node (rank 0) should use values 122 and 321 and pass the difference of the two hashes to the next node.

Like so:

HashInput1 = worldsWorstHash1(122) - worldsWorstHash2(321)
HashInput2 = worldsWorstHash1(HashInput1) - worldsWorstHash2(HashInput1)
HashInput3 = worldsWorstHash1(HashInput2) - worldsWorstHash2(HashInput2)

Print the final two hash values worldsWorstHash1(HashInput3) and worldsWorstHash2(HashInput3) to the console.

Note: Any time a message is sent, output the message to the console.

Expected output:

Rank 0 sending: 134
Rank 1 sending: -100
Rank 2 sending: -330

Result: hash1= -210, hash2= 142

Part 2 (50%)

You are to create an MPI program containing coordinator and participant nodes that will calculate the mean and then calculate the standard deviation of a set of numbers.

The program should contain a *printArray()* method that will print out an array to console in a single line. It should accept two parameters: a pointer to the array and the size of the array.

Include a createArray() function that takes in an integer *n* and returns a random array of size *n*.

Include a *sum()* method that takes in a reference to an array and an array size, and returns the sum of all the values in that array.

Include a *sumDiferences()* method that takes in a reference to an array, an array size, and the overall mean of the dataset. It should produce a sum of the square of differences between each value in the dataset and the mean and return this as the result.

The coordinator and participant methods should do the following:

- Generate an array of random numbers (coordinator only). For predictable results seed the random number generator with 1 and limit the maximum value to 50.
- Determine the size of each partition (coordinator only). Broadcast this to all nodes.
- Scatter the partitions to each node.
- Calculate the mean for this node. Use a reduce operation to gather the overall average.
- Compute the overall average (coordinator only).
- Broadcast the overall average to all nodes and then compute the sum of diferences
- Reduce the overall sum of diferences
- Calculate the standard deviation and print out the dataset, mean and standard deviation (coordinator only).

Modify your code to work with any world size and accept a dataset size from the command line. You may assume that the dataset size will be evenly divisible by the world size.

Report: Perform a comparison evaluating the performance of your program using four nodes against a single node on datasets of different sizes. Try to find a crossover point where the four node version is faster than the single node version. Produce a graph containing this crossover point. Provide a short one page commentary on what this graph states about your algorithm.