GRIFFITH COLLEGE DUBLIN

COMPUTING ASSIGNMENT TITLE SHEET

Course:	BSCH Computing
Stage/Year:	4
Module:	Distributed Systems
Semester:	Semester I
Assignment Number:	Assignment 2
Date of Title Issue:	November 13th, 2024
Assignment Deadline:	Two weeks after submission (see Moodle)
Assignment Submission:	Moodle Upload
Assignment Weighting:	15/50

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

This is an individual assessment to be completed on your own. As such, *the work you submit **must be your own and no one else's.** If you copy from someone else, both parties will be awarded a grade of 0.*

Use of AI tools such as OpenAI or ChatGPT is strictly prohibited.
Avoid the use of any online resources and do not share your code anywhere.

By submitting your assignment you accept that you understand what plagiarism is and that your assignment is not plagiarised in any way.

Any cases of suspected plagiarism will be thoroughly investigated and may be brought in for VIVA.

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
- ❑ Appropriate Method Selection

Assignment 2: Decode and Search

Second Practical Assignment involving the development of a distributed system application to perform the specialised task of data decryption. This requires setting up processes and establishing communication mechanisms to divide and execute tasks in parallel across multiple nodes, enhancing efficiency. Learners are expected to synchronise and coordinate distributed processes to successfully complete the decryption task.

You are required to create **one mpi program**, and **one report.**

Name your files: *FIRSTNAME_LASTNAME_STUDENTNUMBER_PART1.cpp*, *FIRSTNAME_LASTNAME_STUDENTNUMBER_PART2REPORT.pdf*

Submission Format: Zip your submission for Moodle. Name your archive NAME_NUMBER_DSA2.zip. The report must be in **PDF format.** For code include only **.cpp files**, no makefiles or headers.

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.

For each of your programs: Include your name and student number in a comment at the top and print your name and student number <u>once</u> to the console.

Part 1 – MPI Program 60%

You are given a template .cpp file that includes several helper methods: *createData(), decryptText(), searchText(),* and *exportText().* You do not need to modify these methods; simply invoke them as needed. Additionally, you are provided with an individual text file (ciphertext.txt) to be decoded and searched. Save this text file in the same directory as your executable.

Use the provided template to create a distributed decryption MPI program to be run using 5 nodes:

a) The given text file is encrypted with a traditional Caesar cipher using a key from 1-25. The key to be used to decrypt the text file should be taken from the user in console by Node0. The key should then be broadcast to all other nodes.

```
Example: plaintext = ‘abcd’ key = 2 ciphertext = ‘cdef’
```

b) Node0 will read the *ciphertext.txt* file and store the data into an array by invoking *createData()*. The array should then be partitioned into equal sized chunks to be distributed among all nodes.

c) Each node will decrypt their given chunk using the key provided in console using the *decryptText()* function stored into a new char array called *decipheredArray*. After deciphering the text, each node will print its deciphered array.

d) Each node will then invoke *searchText()* on its deciphered partial array to find how many instances of the word "DISTRIBUTED" is contained in its partition. After the function is called, each of the nodes will print its rank and the amount of hits found.

e) Node 1 will collect each of the hits from the multiple nodes and add these together. The result subsequently placed into a new variable, which will be called *totalhits*. Node 2 will output this variable to console in the format: "`Distributed was found a total of <totalhits> times.`"

f) Each node will have a decrypted partial array of differing sizes. Use strlen(partialArray) to store the length of each partial array. Node 0 will collect all the partial decrypted text arrays from each of the nodes, store into *decryptedTotalArray* and export as a text file by invoking the *exportText()* function. Save the text file as *<yourname>DecryptedText.txt* Include in your submission.

Remember to close the MPI library and return control to the OS. Ensure your code will work with a world size of 5.

Part 2 – Documentation and Code Quality 40%

<u>This is a compulsory part of your submission. Any submissions that do not include a completed report will receive an automatic 0.</u>

Use the provided template to complete a report relating to your code for part 1. Answer each question in the report, providing explanations, diagrams and screenshots as requested. Remember to change the Name and Student Number at the top of the report and save as a PDF.

If a diagram is required, you must create it manually using suitable software (e.g., draw.io). Be sure to incorporate your own variable names, data, and other relevant details.

You will also be graded on the quality of your code. Ensure all code you submit is robust, well commented, and well structured. Code that fails to compile will incur a **penalty of 30%.**

Always include a comment at the top of your code with your name and student number, and output these to console once.

Tips:

- ⇒ Keep your code simple: Coordinator() and Participant() methods are not needed.
- ⇒ Read the brief, then read it again. Then read the brief.
- ⇒ Ensure your code is tidy and readable.
- ⇒ Include appropriate comments.
- ⇒ Make sure your code compiles and runs correctly.
  - Anything not working, comment out and tell me why.
- ⇒ Information that you output to console should be clear and concise.
  - For example, your name/number should only be printed once
- ⇒ Use the most appropriate MPI function for the task.
- ⇒ Use the provided template and helper methods.
- ⇒ Avoid '`using namespace std;`', use instead `std::` before any use of the library such as `cout` or string etc.
- ⇒ You will be graded on the quality of your code.