# Assignment Brief 2 (RQF)

## Higher National Certificate/Diploma in Business

|  |  |
| --- | --- |
| **Student Name/ID Number:** |  |
| **Unit Number and Title:** | Unit 19: Data Structures and Algorithms |
| **Academic Year:** | **2021** |
| **Unit Assessor:** |  |
| **Assignment Title:** | Implement and assess specific DSA |
| **Issue Date:** |  |
| **Submission Date:** |  |
| **Internal Verifier Name:** |  |
| **Date:** |  |

|  |
| --- |
| **Submission Format:** |
| *Format:*   * The submission is in the form of an individual written report and a presentation. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system.   *Submission*   * Students are compulsory to submit the assignment in due date and in a way requested by the Tutor. * The form of submission will be a soft copy posted on <http://cms.greenwich.edu.vn/>. * Remember to convert the word file into PDF file before the submission on CMS.   *Note:*   * The individual Assignment *must* be your own work, and not copied by or from another student. * If you use ideas, quotes or data (such as diagrams) from books, journals or other sources, you must reference your sources, using the Harvard style. * Make sure that you understand and follow the guidelines to avoid plagiarism. Failure to comply this requirement will result in a failed assignment. |
| **Unit Learning Outcomes:** |
| **LO3** Implement complex data structures and algorithms  **LO4** Assess the effectiveness of data structures and algorithms |
| **Assignment Brief and Guidance:** |
| **Assignment scenario**  Continued from Assignment 1.  **Tasks**  For the middleware that is currently developing, one part of the provision interface is how message can be transferred and processed through layers. For transport, normally a buffer of queue messages is implemented and for processing, the systems requires a stack of messages.  The team now has to develop these kind of collections for the system. They should design ADT / algorithms for these 2 structures and implement a demo version with message is a string of maximum 250 characters. The demo should demonstrate some important operations of these structures. Even it’s a demo, errors should be handled carefully by exceptions and some tests should be executed to prove the correctness of algorithms / operations.  The team needs to write a report of the implementation of the 2 data structures and how to measure the efficiency of related algorithms. The report should also evaluate the use of ADT in design and development, including the complexity, the trade-off and the benefits. |

|  |  |  |
| --- | --- | --- |
| Learning Outcomes and Assessment Criteria (Assignment 2) | | |
| Pass | Merit | Distinction |
| **LO3** Implement complex data structures and algorithms | | **D3** Critically evaluate the complexity of an implemented ADT/algorithm |
| **P4** Implement a complex ADT and algorithm in an executable programming language to solve a well defined problem.  **P5** Implement error handling and report test results. | **M4** Demonstrate how the implementation of an ADT/algorithm solves a well-defined problem |
| **LO4** Assess the effectiveness of data structures and algorithms | | **D4** Evaluate three benefits of using implementation independent data structures |
| **P6** Discuss how asymptotic analysis can be used to assess the effectiveness of an algorithm  **P7** Determine two ways in which the efficiency of an algorithm can be measured, illustrating your answer with an example. | **M5** Interpret what a trade-off is when specifying an ADT using an example to support your answer |

*Tree Application:*

* Build a question tree:
  + Run program until get the final answer (look like Decision tree)
* Build a directory tree;
  + Input: search a folder name -> output: return String path.
* Auto-complete feature: Heap, BST, Tree.
* Huffman Coding

Stack Application:

* Matching Parentheses

Queue Application:

* Process a message queue

# Implement ADT & algorithms

## Description of [your application]

Describe your application to be implemented, describe the reason that you choose a data structure.

*Hint:*

*Choose one problem (well-defined problem) to solve. Describe it (detail) and give an idea to solve it.*

*Introduction*

* *Why do you choose this problem?*
* *What do you choose the complex ADT to handle this issue? Why ?*

## ADT (P4, P5)

### Implement

* Explain your implementation of the data structure (P4)
  + *Implementing of complex ADT (p/s: a brief explanation of the functions in each class/interface)*
    - *Screen shoots of your code*
  + *Using this ADT to solve above problem (+main function))*
    - *Screen shoots of your main and console (with description)*
* Explain how you handle error by exception (P5)
  + *What is error handling? Definition of Try-catch block?*
  + *The benefits of error handling and exception Handling?*
  + *Report test results: Show test cases table and test results (automatic Junit or manual test). Choose one:*
    - *All methods of your ADT*
    - *Your algorithms which solve the well-defined problem in P4*
* Explain how do you use the data structure in your whole application (M4)

*(illustration + Explanation)*

### Test (P5)

* Show test cases table and test results (automatic Junit or manual test) (P5)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input data | Expected output | Actual output | Result |
| Test name | Input data | Expected output | Actual output | Pass/Fail |
|  |  |  |  |  |

* Screen shoots of running the tests above (for Actual output)
* Why use Custom Exception? = the benefits of user-defined exceptions
* Custom exception extend RunTimeException -> RuntimeException...
* Code...

# Analysis

## Big O (P6)

* Explain about Big O with some simple examples (theory)

*Hint:*

1. *- What is asymptotic analysis?*

*- What is best, average and worst case in an algorithm?*

*- How to analyze algorithms?*

*- What is time complexity?*

*How to evaluate time complexity?*

*-Sequential Statements*

*Int a;*

*a=5+6;*

*-Conditional Statements*

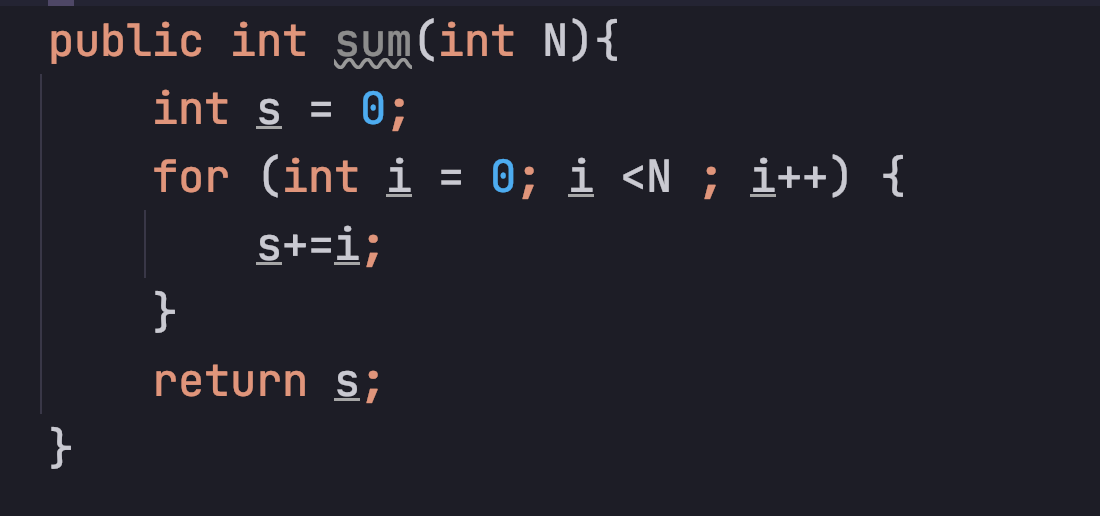
*If(a==0)*

*Return true;*

*Else return false;*

*-Loop Statements*

*Example (A)*

**

|  |  |
| --- | --- |
| *int s =0* | *1* |
| *int i=0* | *1* |
| *i<N* | *N+1* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*- What is space complexity?*

*How to evaluate space complexity? Space complexity = Space used input values + Auxiliary Space*

*Java -> “Primitive data types size in Java” => option 1: table (bytes); option 2: introduction in Oracle website (bit)*

*Example (B)*

*-* ***What is Growth rate****?*

*Show some common growth rates to compare.*

1. *The importance of asymptotic analysis?*
2. *What are asymptotic notations?*

*Big(O) -> worst*

*Big(Omega) -> best*

*Big(Theta) -> Average*

*Give an example for each one.*

## Analyze performances (P7, D3)

* Evaluate some of your important algorithmS in task 1 by time (P7) and by Big O (D3) ((O(1), (O(n), O(n2), …).

Must have

*Hint:*

*Applying asymptotic analysis*

*(Just for time complexity)*

*Pick two of my lab (17\_Lab4\_Algorithm Analysis\_Student).*

*P7: evaluate T(n). Explain how can you get that result (T(n))*

*D3: evaluate T(n) -> O(n). Explain how can you get that result (O(n))*

## Discussion (M5, D4)

* Advantages & disadvantages when implement ADTs and use them in your application

M5:

a. *Time-space complexity Trade-off*

*1. What is time-space complexity trade-off?*

*2. Types of Time-space Trade-off. Give an example for each one.*

*- Compressed or Uncompressed*

*- Stored Image or Re render*

*- Smaller code or Loop unrolling*

*- Look up table or Recalculaltion*

*b. Example about time-space complexity trade-off*

*you can choose an algorithm which can be solved by two ways (iterative and recursive).*

*Discussion about the trade-off? What? Why?*

D4:

*List three benefits of using implementation independent data structures and explain in detail.*