# Luminus Technical University College - Assignment Brief (RQF)

## Higher National

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Student Name** | | **Rashed Hasan Qahah** | | | **Language of assessment** | | | **AR** | **EN** |
| **College ID:** | | |  | |
| **Pearson ID:** | | |  | |
| **Unit Number and Title** | |  | **19. Data Structures and Algorithms** | | | | | | |
| **Academic Year** | |  | | | | | | | |
| **Unit Tutor** | |  | | | | | | | |
| **Internal Verifier Name and Approval (Signature)** | |  | | | | **Approval Date:** | | | |
|  | | | |  | | | |
| **Assignment number and Title** | | **1** | **Design and implement complex data structures and algorithms** | | | | | | |
| **Issue Date (1St Submission)** | |  | | **Submission Date (1st Submission)** | | |  | | |
| **Issue Date (2nd Submission)** | |  | | **Completion Date (2nd Submission)** | | |  | | |
| **Submission Format** | | | | | | | | | |
| **Code file, word document at least 1000 words, use consistent style and formatting.**  **Authenticity**: Your proposed work should be authentic, and not copied from others. your answers have to relate to the vocational scenario and not generally. 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.** | | | | | | | | | |
| **Unit Learning Outcomes** | | | | | | | | | |
| **LO1** | **Examine abstract data types, concrete data structures and algorithms.** | | | | | | | | |
| **LO2** | **Specify abstract data types and algorithms in a formal notation.** | | | | | | | | |
| **LO3** | **Implement complex data structures and algorithms.** | | | | | | | | |
| **LO4** | **Assess the effectiveness of data structures and algorithms.** | | | | | | | | |
| Transferable skills and competencies developed | | | | | | | | | |
| This assignment provides practice on designing and building data structures and utilize them for building systems according to certain requirements.  Also, it teaches students to compare different algorithms and assess their effectiveness.  Along with the following:   * identify program data requirements. * specify abstract data types using formal notation. * translate into concrete data structures and be able to develop, using a programming paradigm, different sorting, searching and navigational algorithms that implement complex data structures and evaluate their effectiveness. | | | | | | | | | |
| **Vocational scenario:** | | | | | | | | | |
| You are working in a Tech Solutions company that designs and builds software solutions for several clients.  Your company has won the bid to create an application that handle many required services for MacDonalds. an application for handling different features for MacDonalds services.  Consider the tasks below to implement the system. | | | | | | | | | |
| Assignment activity and guidance | | | | | | | | | |
| The application has many features as listed and explained in the following tasks:  Task 1: Handling Orders  Description: cars in drive thru line place their order once it is their turn. The orders are delivered in a first-come-first-serve order.  1.1: Draw a class diagram that includes the following classes and show the relationship between them, information hiding, and ADT specifications:  **Order:** each order placed from drive thru service contains: order ID, l**ist of items**, total price. This class have an operation to calculate the total price based on items (not user input), and a print operation.  **Item:** each item has a description and a price. For example, (Big Mac, 5.00)  **Queue:** implement over a DLL to handle orders. With main add and remove operations of a queue.  **DLL:** customize the DLL class to store orders, and another one to store items.  The DLLOrder class has two operations, AddOrder that add an order to the end of the list, and a RemoveOrder that return the removed order from the head/front.  The DLLItems has only one operation (AddItem) that add an Item to the end of the list.    **1.2:** Implement the solution according to the given classes and specifications, and prevent handling orders when there are no orders in queue and prevent creating invoice/total price for empty list of items .  **1.3:** Test your implementation using a Demo class, Figure 1 shows a sample testing. **Present** how the specified ADT solves the given problem.    **Figure 1 : sample testing**  **1.4:** **Draw** an illustration to show how queue handles the orders in FIFO manner, give examples.  **1.5:** Consider implementing the Queue class over an array, and SLL. **Discuss** the implementation details in term of space and time complexities as well as the possible trade-offs regarding queue specifications (enqueue and dequeue).  **1.6:** In Mac they have a job to put plates in a cabinet. The way they put the plates is one on top of another until the cabinet is full. When they want to use the plates, they take the plate that is on top in the cabinet until they are all finished. They have brought a robot to do the job. Help them by **defining** a stack specification so they can use it to implement the solution, **show** how stack handle the solution.  **Task 2: ADT and OOP**  **2.1:** You have implemented the above solution using the Object-Oriented Programming principles. **Negotiate** the idea that ADTs are the basis of object orientation. Justify your answer whether you agree or disagree.  **2.2:**  **Explore** the advantages of encapsulation and information hiding, class diagram representation when using the queue ADT.  **2.2:** **Assess** three benefits of implementing independent Data Structure such as queue.  **Task 3: Testing sorting algorithms to improve orders handling**  You have been asked to implement a feature to sort the orders before adding them to the queue based in their total price. This will give orders with highest price the priority to be handled first. For the easiness of implementation and demonstration, assume that you are given an array of total prices. Sort it in **descending** order using two sorting algorithms (merge sort and insertion sort). And **compare** their performances using time benchmarking technique over three different shapes of arrays.  Use the following table to organize your results, N is array size:   |  |  |  |  | | --- | --- | --- | --- | | **N= 100000** | **Sorted** | **Reversely sorted** | **Random** | | **Merge sort** | **Time in nanoseconds** |  |  | | **Insertion sort** |  |  |  |   **3.1:** Consider how asymptotic analysis is used to evaluate the effectiveness of algorithms. Use the sorting techniques used as an example.  **3.2:** You have measured both sorting algorithms efficiency using time benchmarking technique. **Determine** other two ways to measure the algorithm efficiency, support your answer with examples.  **3.3:** **Discuss the performance** both behaviours of algorithms in relation to their implementation, complexity, array sizes, different array shapes, and evaluate the obtained outcomes against the intended ones.  Use the following code to measure time in java:    **Figure 2: How to measure time of your program in java**  Task 4: Adding a delivery feature  You have given the following map of different branches of MacDonalds in your city. And asked to implement another feature to food delivery service.    Figure 3: MacDonalds Map, b: branch, c: client  4.1: Define what is the closest MacDonalds branch for the delivery man to pick the order from for each client. Show detailed tracing using *bellman ford* and *Dijkstra's* algorithm shortest path algorithms.  4.2: Critically analyse the complexity of both algorithms in reference to their DS implementation. | | | | | | | | | |
| **Recommended Resources**  **Please note that the resources listed are examples for you to use as a starting point in your research – the list is not definitive.**  **Textbooks** | | | | | | | | | |

**Learning Outcomes and Assessment Criteria**

|  |  |  |
| --- | --- | --- |
| Pass | Merit | Distinction |
| LO1 Examine abstract data types, concrete data structures and algorithms | |  |
| P1 Create a design specification for data structures explaining the valid operations that can be carried out on the structures.  P2 Determine the operations of a memory stack and how it is used to implement function calls in a computer. | M1 Illustrate, with an example, a concrete data structure for a First In First out (FIFO) queue.  M2 Compare the performance of two sorting algorithms | D1 Analyse the operation, using illustrations, of two network shortest path algorithms, providing an example of each.  D2 Discuss the view that imperative ADTs are a basis for object orientation and, with justification, state whether you agree. |
| LO2 Specify abstract data types and algorithms in a formal notation | |
| P3 Using an imperative definition, specify the abstract data type for a software stack. | M3 Examine the advantages of encapsulation and information hiding when using an ADT |

**Learning Outcomes and Assessment Criteria**

|  |  |  |
| --- | --- | --- |
| Pass | Merit | Distinction |
| LO3 Implement complex data structures and algorithms | |  |
| 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 | D3 Critically evaluate the complexity of an implemented ADT/algorithm.  D4 Evaluate three benefits of using implementation independent data structures. |
| LO4 Assess the effectiveness of data structures and algorithms | |
| 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. |