جامعة الحسين التقنية
Al Hussein Technical University

ASSIGNMENT BRIEF

HTU Course No: 40201201	**HTU Course Name:** Data Structures & Algorithms
BTEC Unit Code: T/618/7430	**BTEC UNIT Name:** Data Structures & Algorithms

Student Name/ID Number/Section	
HTU Course Number and Title	40201201 Data Structures & Algorithms
BTEC Unit Code and Title	T/618/7430 Data Structures & Algorithms
Academic Year	2024-2025 1
Assignment Author	Balqis Aldabaibeh
Course Tutor	Balqis Aldabaibeh - Zaineh yousef
Assignment Title	Using Data Structures and Algorithms to Build Medical Managment System
Assignment Ref No	1
Issue Date	21/11/2024
Formative Assessment dates	From 21/11/2024 to 09/01/2025
Submission Date	27/01/2025
IV Name & Date	Ahmed Bataineh 20/11/2024

Submission Format

In this assignment, you are required to submit:

- An individual written report; for all tasks.
- Signed Declaration Form.
- A full implemented and functioning code; for tasks that require implementation.
- An oral demonestration with your instructor of your submitted word

General Guidelines:

- The report should be well-formatted and in a Word version. Clearly state the answers by question part and subtask numbers.
- All implemented ADTs and Algorithms should be fully implemented.
- All submissions should be uploaded to the university LMS (the eLearning platform) within the stated time and date.
- NO EMAIL SUBMISSIONS OR LATE SUBMISSIONS WILL BE ACCEPTED.
- HTU policies and regulations will be applied for any kind of plagiarism.
- The oral demonestration will include everything submitted in the written report, debugging and evaluating the submitted code, and the material basic concepts delivered during classes.
- The attendance of the oral demonestration is mandatory, the exact date and time will be determined by your instructor and announced later .

Unit Learning Outcomes

LO1 Examine abstract data types, concrete data structures and algorithms

LO2 Specify abstract data types and algorithms in a formal notation

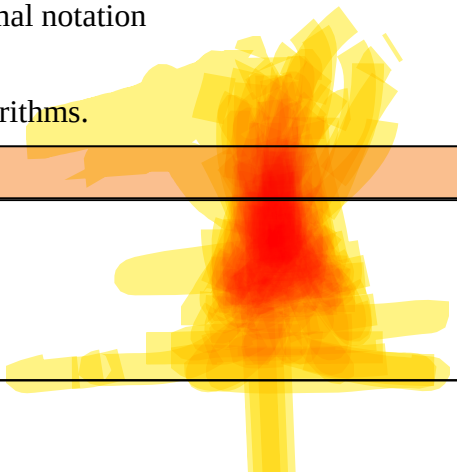
LO3 Implement complex data structures and algorithms

LO4 Assess the effectiveness of data structures and algorithms.

Assignment Brief and Guidance

Medical Management System

Vocational scenario



The Medical Management System is designed to streamline and efficiently manage various hospital operations. The system handles patient registration, appointments, discharge management, and emergency. The system manages various tasks related to patient check-ins, appointments, and emergency prioritization in a hospital. It integrates different data structures such as **Linked List**, **Queue**, **Stack**, **Graph**, and **Binary Search Tree (BST)** to facilitate the operation of the hospital. Below is an explanation of each feature and how it can be implemented using the appropriate data structures:

1. **Patient Registration using Linked List:**
   - Register a patient, storing their name, ID, age, and ailment.
   - Discharge management: When patients are discharged, they are removed from the linked list, ensuring only active patients are in the system.
2. **Regular Appointments Management in the Waiting Room using Queue:**
   - Manage a list of non-emergency appointments. Patients are seen by the doctor in a first-come, first-served (FCFS) order. Patients waiting for regular appointments are added to the queue, and the doctor processes them sequentially, in the order they arrive.
3. **Tracking Patient Discharge and Re-Admission using Stack:**
   - When a patient is temporarily discharged (e.g., for testing or consultation), their patient ID is pushed onto a stack.
   - When re-admission is required, the patient ID is popped from the stack to bring them back into the system.
   - This Last-In-First-Out (LIFO) structure ensures patients are re-admitted in the reverse order of discharge.
4. **Sorting Patients by Age and ID**
   - Helps in generating sorted lists of patients, useful for managing large hospitals with many patients needing frequent sorting of records
   - **Algorithms**: Different sorting algorithms (e.g., **Algorithm A** for ID, **Algorithm B** for age) can be implemented depending on the sorting criteria.
5. **Finding The Closest Exit in Emergency Situations using Graphs**
   - In the case of an emergency, patients and staff need to be directed to the nearest exit.
   - The hospital layout can be represented as a graph, where rooms, hallways, and exits are nodes, and paths between them are edges.
   - The system uses **shortest-path algorithms** (e.g., **Dijkstra's Algorithm**) to calculate and highlight the nearest exit, guiding the crowd quickly and efficiently.
6. **Administration Management and Employee Search using BST.**
   - For managing the administration, the system maintains a BST of employees
   - The BST allows efficient searching for employees by name, ID, or other attributes.

Part 1- Building the Medical System:

- 1. Provide a class diagram for the major manager class and basic ADT used in building the system, explaining each valid operation in the text. (**Report**).
- 2. Explain the stack ADT used in your design to solve feature 3, and what its valid specifications. Provide an example that illustrates its definition. (**Report**)
- 3. Implement the proposed system functionality using JAVA programming language using your own implementation of all different ADTs. Ensuring error handling. Provide evidence of at least two error-handling cases. Figure 1 provides a hint of how BST stores employee IDs(**Report + code**).

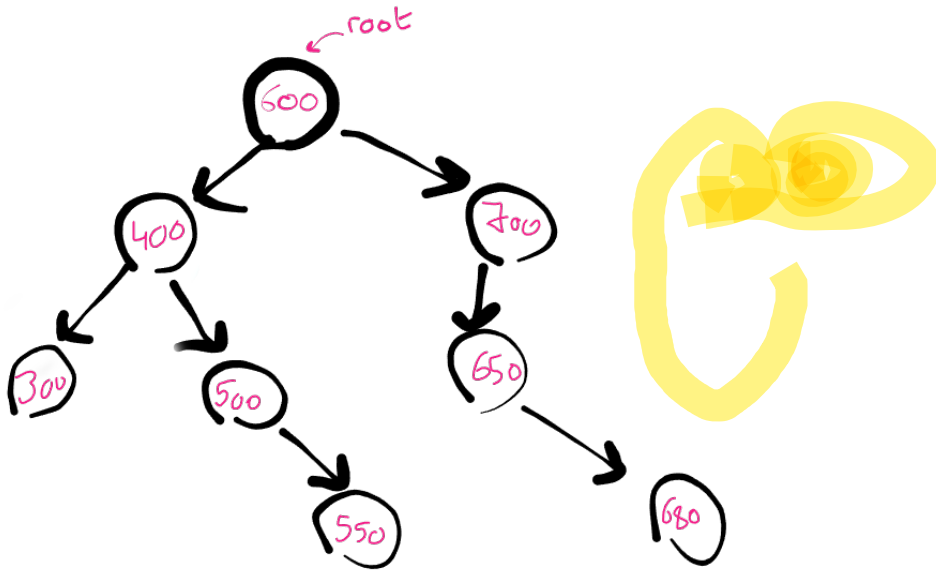


Figure 1: sample of BST that stores employee IDs

- 4. For feature 4 in the system apply different sorting algorithms for each sorting criterion. Select from these you study in class (selection sort, insertion sort, bubble sort, merge sort). Compare both sorting algorithms in terms of best case and worst case. Experiment with different sizes of patient numbers and register the time required in nanoseconds. (**Report + code**).
- 5. Assume that the queue used for implementation feature 2 is built over a circular array or doubly linked list. Illustrate (by drawing) the different concrete implementation details for a sequence of enqueue and dequeue operations between a circular array and a doubly linked list. (**Report**).
- 6. Suggest a real-world test scenario that demonstrates each operation/functionality of the system, and discuss how the specified ADT helps in solving this functionality. (**Report**)
- 7. Illustrate the operations of two shortest path algorithms bellman-ford and Dijkstra algorithms using step by step. Using the example implemented in Feature 5. Figure 2 gives you an idea of how to represent the graph(**Report + code**).

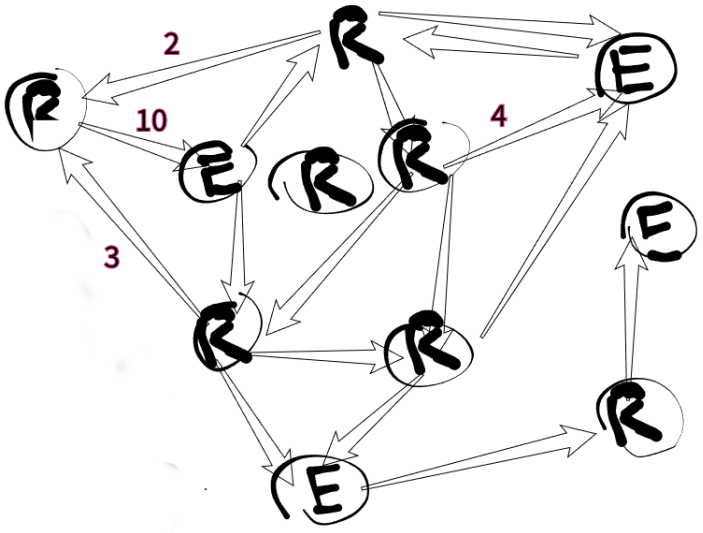


Figure 2: Sample of connected Graph consists of a set of Rooms and Exists. Notice that some edges has weight for demonstration purposes.

- 8. Provide a critical evaluation of all algorithms used in the implementation of your solution, considering searching, sorting, shortest path algorithms, and others. (**Report**).
- 9. For feature 6, discuss the trade-off of this feature implementation consider using Sorted Array and BST ADTs. In terms of time complexity and best and worst cases if occurs. (**Report**).

Part 2-Concepts:

- 1. Given the code in Figure 3, and the data in Figure 4. Show how the stack is used to handle the function calls by showing its content step by step through code execution in this task. Determine the action needed for function execution and map them to different stack operations. (**Report**).

```
1  public void print(Node node) {
2    if (node == null) {
3      return;
4    }
5    print(node.next);
6    System.out.print(node.data + " ");
7  }
```

Figure 3: a piece of code to be executed on data in Fig 4

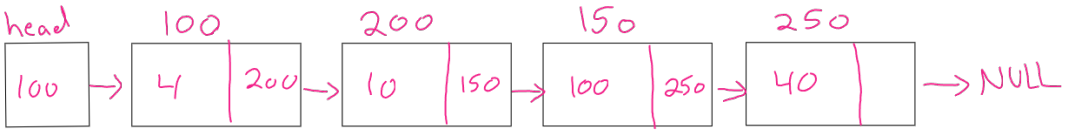


Figure 4: Instance of SLL

- 2. By example, explain how the asymptotic analysis is critical to assess the effectiveness. Compare two complexities and explain how, for smaller input sizes, the performance might differ without considering Big-O notation, even though the long-term behavior remains consistent (**Report**).
- 3.How the algorithm efficiency can be measured, list two ways, and give an example.(**Report**).
- 4.You are trying to shop while keeping within a specific budget, so you want to track the total cost of items as you add them to your basket. Check if adding a new item will exceed your budget, remove items if no longer needed or they are costly, and check the total cost, and check the list of selected items before the final checkout. Consider an ADT called **Basket** with the following specifications (represented in class diagram) to help you with the solution: (**Report**).

Basket
-items: List<String> -prices: List<Double> -budget: double -totalCost: double
+ setBudget(budget: double): void + addItem(item: String, price: double): boolean + removeItem(item: String): boolean + getTotalCost(): double + getRemainingBudget(): double + listItems(): List<String> + isWithinBudget(): boolean

- You are required to provide a JAVA main code. Or high-level pseudo code for your solution.
- Explain using your provided solution what advantages of encapsulation and information hiding help you in using **Basket** ADT for solving a given problem.

- 5.Explain how the core principles of OOP align with the goals of designing and managing an ADT.(**Report**).
- 6.Evaluate three benefits of using independent data structures in your medical system, understand the benefits, discuss the following questions.(**Report**).

  - Why is it beneficial to use a **queue** or **stack** in a **medical management system** instead of directly using arrays or other basic data structures?
  - What are the main differences between a **queue** and a **stack**? In what scenarios would you choose one over the other in a medical management system?
  - Explain how using implementation-independent data structures like stacks and queues can improve the maintainability and scalability of your medical management system.
  - How would you write test cases for your queue implementation to ensure its correctness? What are the edge cases you would test for in a medical management system?

Learning Outcomes and Assessment Criteria			
Learning Outcome	**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.
LO2 Specify abstract data types and algorithms in a formal notation	**P3** Specify the abstract data type for a software stack using an imperative definition.	**M3** Examine the advantages of encapsulation and information hiding when using an ADT.	**D2** Discuss the view that imperative ADTs are a basis for object orientation offering a justification for the view.
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.
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.	**D4** Evaluate three benefits of using implementation independent data structures.