Higher Nationals in Computing

Unit 19: Data Structures and Algorithms

ASSIGNMENT 1

Assessor name: **PHAN MINH TAM**

Learner's name:

ID:

Class:

Subject code: 1649

Assignment due: Assignment submitted:

ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 19: Data Structures and Algorithms		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name		**Student ID**	
Class		**Assessor name**	Phan Minh Tam

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	

Grading grid

P1	P2	P3	M1	M2	M3	D1	D2

☼ Summative Feedback:		☼ Resubmission Feedback:
Grade:	Assessor Signature:	Date:
Internal Verifier's Comments:		
Signature & Date:		

ASSIGNMENT 1 BRIEF

Qualification	**BTEC Level 5 HND Diploma in Business**		
Unit number	Unit 19: Data Structures and Algorithms		
Assignment title	Examine and specify ADT and DSA		
Academic Year	2022		
Unit Tutor	TamPM		
Issue date	12 – September – 2022	**Submission date**	
IV name and 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:
LO1 Examine abstract data types, concrete data structures and algorithms **LO2** Specify abstract data types and algorithms in a formal notation
Assignment Brief and Guidance:

<u>Assignment scenario</u>

You work as in-house software developer for Softnet Development Ltd, a software body-shop providing network provisioning solutions. Your company is part of a collaborative service provisioning development project and your company has won the contract to design and develop a middleware solution that will interface at the front-end to multiple computer provisioning interfaces including SOAP, HTTP, JML and CLI, and the back-end telecom provisioning network via CLI .

Your account manager has assigned you a special role that is to inform your team about designing and implementing abstract data types. You have been asked to create a presentation for all collaborating partners on how ADTs can be utilised to improve software design, development and testing. Further, you have been asked to write an introductory report for distribution to all partners on how to specify abstract data types and algorithms in a formal notation.

Tasks

Part 1

You will need to prepare a presentation on how to create a design specification for data structures, explaining the valid operations that can be carried out on the structures using the example of:

1. A stack ADT, a concrete data structure for a First In First out (FIFO) queue.
2. Two sorting algorithms.
3. Two network shortest path algorithms.

Part 2

You will need to provide a formal written report that includes the following:

1. Explanation on how to specify an abstract data type using the example of software stack.
2. Explanation of the advantages of encapsulation and information hiding when using an ADT.
3. Discussion of imperative ADTs with regard to object orientation.

Learning Outcomes and Assessment Criteria (Assignment 1)		
Pass	**Merit**	**Distinction**
LO1 Examine abstract data types, concrete data structures and algorithms		**D1** Analyse the operation, using illustrations, of two network shortest path algorithms, providing an example of each.
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.	
LO2 Specify abstract data types and algorithms in a formal notation		**D2** Discuss the view that imperative ADTs are a basis for object orientation and, with justification, state whether you agree.
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.	

Table of Contents

ASSIGNMENT 1 ANSWERS

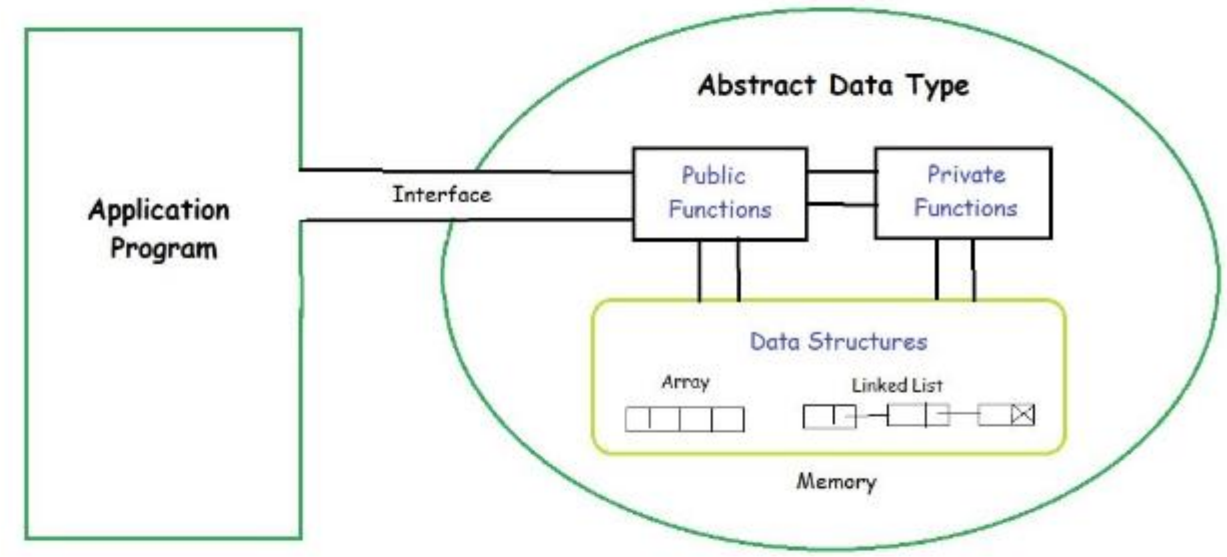
P1 Create a design specification for data structures explaining the valid operations that can be carried out on the structures.

1. Introduction to Abstract Data Type (ADT).

1.1 What is ADT.

An abstract data type (ADT) is a class or type for an object whose behavior is determined by a set of values and operations.ADT's definition only specifies the operations to be carried out, not how they will be carried out. It doesn't say what algorithms will be used to carry out the operations or how the data will be organized in memory. Because it presents an implementation-independent viewpoint, it is referred to as "abstract."

Abstraction is the process of providing only the essentials and concealing details.



https://www.geeksforgeeks.org/abstract-data-types/

1.1.2 Example of ADT.

An ordered collection of items can be accessed using an integer index in an array, a fundamental abstract data type.These items can be simple types like integers or more complicated types like classes' instances.Since it is an ADT, it does not specify a method of implementation; however, an array (data structure) or dynamic array is almost always used.

A resizable array is the ArrayList class in the java.util package.In Java, the size of an array cannot be changed; users must create a new array in order to add or remove elements. This is the difference between a built-in array and an arraylist.In contrast, elements in an arraylist can be changed at any time.

No	Operator	Name	Type	Description
1	Void add (int item)	Add at the end	Int -> void	Adding anew element to end of the list. A = [1, 3, 5, 7]
2	Void add (int pos, int item)	Add	Int -> void	Adding a new element at position pos in the List and moving elements that were previously in positions pos through size() - 1 one position to the right to make room (returning error when pos is less than 0 or larger than size()). A = [1, 3, 5, 7] **Add** (1, 2);->A = [1, 2, 3, 5, 7]
3	Boolean contains (int item)	Find	Int -> bool	Returning true indicates that the element is present in the List (that is, there is an item x in the List that is equivalent to item). A = [1, 2, 3, 5, 7] **Contains(0)**;->false **Contains(1)**;->true
4	Int size()	Return size	Int->int	Returning the number of

				elements in the List. A = [1, 2, 3, 5, 7] **Size**();->5
5	Boolean is Empty()	Check empty	Int->int	If the ArrayList is empty, return true. A = [1, 2, 3, 5, 7] **IsEmpty()**;->false
6	Int get (int pos)	Get	Int->int	Returning the List element at position pos (erroneous if pos is less than or equal to size()). A = [1, 2, 3, 5, 7]->**get(0)**;->1
7	Int remove (int pos)	Removef	Int->int	Moving elements that were previously in positions pos + 1 through size() -1 one position to the left in order to fill in the gap (returning an error if pos is smaller than 0 or larger than or equal to size()), removing the element from position pos in the ArrayList, and returning it

1.1.2 List ADT.

• The information is by and large put away in key grouping in a rundown which has a head structure comprising of count, pointers and address of contrast capability required with look at the information in the rundown.

• The self-referential pointer to the next node in the list and the pointer to a data structure are both contained in the data node.

• The following is a list of the ADT functions:

• Get an element from the list at any position using get().

• Insert an element at any point in the list with the function insert().

• Remove() removes the first time any element appears in a list that is not empty.

• RemoveAt() removes an element from a non-empty list at a predetermined location.

• Replace an element at any position with another element using replace().

• Size() returns the list's number of elements.

• IsEmpty(): If the list is empty, returns true; otherwise, returns false.

• IsFull(): If the list is full, returns true; otherwise, returns false.

1.1.3 Stack ADT.

• The pointer to the data is stored in the Stack ADT implementation rather than the data itself.

• The address is sent to the stack ADT after the program allocates memory for the data.

• The ADT contains both the data nodes and the head node.The only thing the calling function can see is the stack's pointer.

• A count of the number of entries currently in the stack and a pointer to the top are also contained in the stack head structure.

• push() adds an element known as top to one end of the stack.

• If the element at the top of the stack is not empty, pop() will remove it and return it.

• If the stack is not empty, peek() returns the element at the top of the stack without removing it.

• Size() returns the number of stack elements.

• IsEmpty() - Return valid in the event that the stack is unfilled, in any case get back bogus.

• IsFull(): If the stack is full, returns true; otherwise, returns false.

1.2 The benefits of ADT.

REFERENCES

Harvard Reference List Citations for Books:

Last name, First initial. and Last name, First initial. (Year published). Title. City: Publisher, Page(s).

Examples:

- Desikan, S. and Ramesh, G. (2006). *Software testing*. Bangalore, India: Dorling Kindersley, p.156
- Daniels, K., Patterson, G. and Dunston, Y. (2014). *The ultimate student teaching guide*. 2nd ed. Los Angeles: SAGE Publications, pp.145-151

Harvard Reference List Citations for Websites

Last name, First initial (Year published). Page title. [online] Website name. Available at: URL [Accessed Day Mo. Year].

Website name, (Year published). *Page title*. [online] Available at: URL [Accessed Day Mo. Year].

Examples:

- Messer, L. (2015). *'Fancy Nancy' Optioned by Disney Junior*. [online] ABC News. Available at: http://abcnews.go.com/Entertainment/fancy-nancy-optioned-disney-junior-2017/story?id=29942496#.VRWbWJwmbs0.twitter [Accessed 31 Mar. 2015].
- Mms.com, (2015). *M&M'S Official Website*. [online] Available at: http://www.mms.com/ [Accessed 20 Apr. 2015].