

ASSIGNMENT 1 FRONT SHEET

Qualification	BTEC Level 5 HND Diploma in Computing		
Unit number and title	Unit 19: Data Structures and Algorithms		
Submission date	28/06/2024	Date Received 1st submission	
Re-submission Date		Date Received 2nd submission	
Student Name	Nguyen Duc Tu	Student ID	GCH200690
Class	RE SU24	Assessor name	Do Hong Quan
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:**

2.1

Grade:
Assessor Signature:
Date:
Internal Verifier's Comments:
IV Signature:

Table of Contents

Introduction.	4
Abstract Data Types (ADTs).	4
Figure 1 - 2: Linked list.	4
Stack ADT.	5
Figure 3: Push()	5
Figure 4: Peek()	5
Figure 5: Pop().....	5
Figure 6: Input: LIVE -> EVIL.....	6
Memory Stack.	6
Figure 7 - 8 - 9: Memory stack operation.	7
Queue ADT.....	8
Figure 10: Queue.	8
Explanation on how to specify an abstract data type using the example of software stack.	8
References.	10

Introduction.

This assignment is about teaching the author's team about abstract data types (ADTs). The author will give a presentation explaining how ADTs can improve software design and also create a report for everyone that explains how to define ADTs and algorithms.

The first part of the assignment involves designing data structures and explain what operations can be done on these structures.

Abstract Data Types (ADTs).

Definition: An ADT is a blueprint for a data type. It focuses on how the data is used from a user's perspective, not how it's stored internally. This is different from data structures, which are the actual way data is stored in the computer (GeeeksforGeeks, 2023).

An ADT has three key parts:

1. **Data.**
2. **Operations.**
3. **Encapsulation of data and operations:** The data is hidden from the user and can only be accessed through the defined operations. Users don't need to know how the data is stored, just how to use it.

In this assignment, the author will define two specific ADTs: stacks and queues.

This is examples of abstract data type:

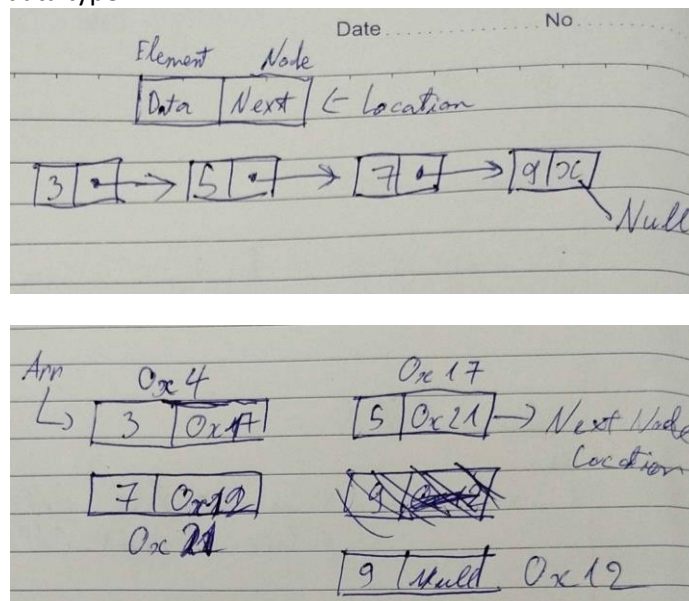


Figure 1 - 2: Linked list.

"Next" is the location in RAM.

Stack ADT.

According to GeeksforGeeks, 2023. A stack is a simple data structure that organizes items in a specific order, like a stack of plates. New items are added on top (push), and the first item to be removed is always the one that was added last (pop). This is called LIFO (Last In, First Out).

Here are the basic operations you can perform on a stack following Oracle, 2022:

- **isEmpty()**: Checks if the stack has any items.
- **push(item)**: Adds a new item to the top of the stack.
- **peek()**: Lets you see the top item without removing it.
- **pop()**: Removes and returns the top item from the stack.

Push:

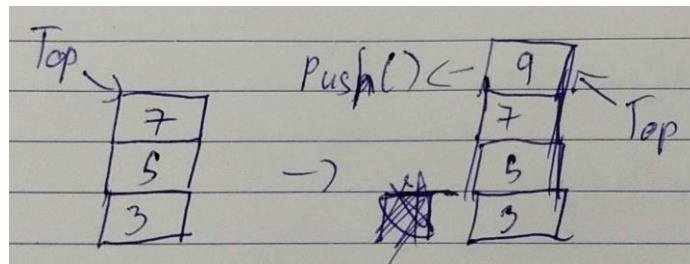


Figure 3: Push()

Peek:

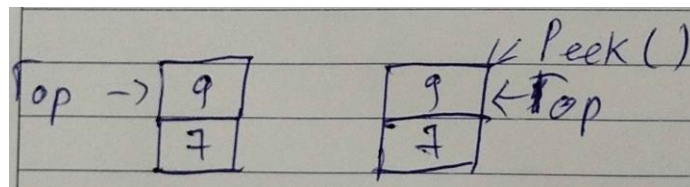


Figure 4: Peek()

Pop:

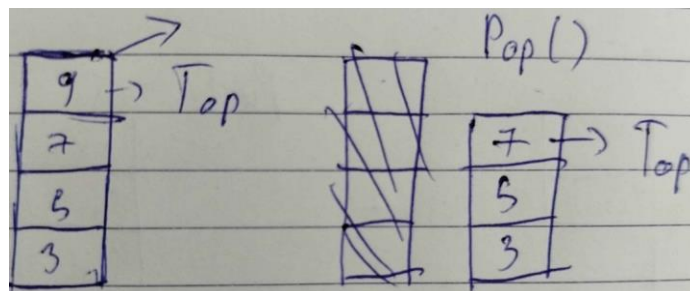


Figure 5: Pop()

Example:

You can reverse a string using a stack. Imagine the string as a stack of plates. You can reverse the order by adding (pushing) each character onto the stack one by one. Then, you can take the characters out (popping) in the opposite order, which will be the reversed version of the original string.

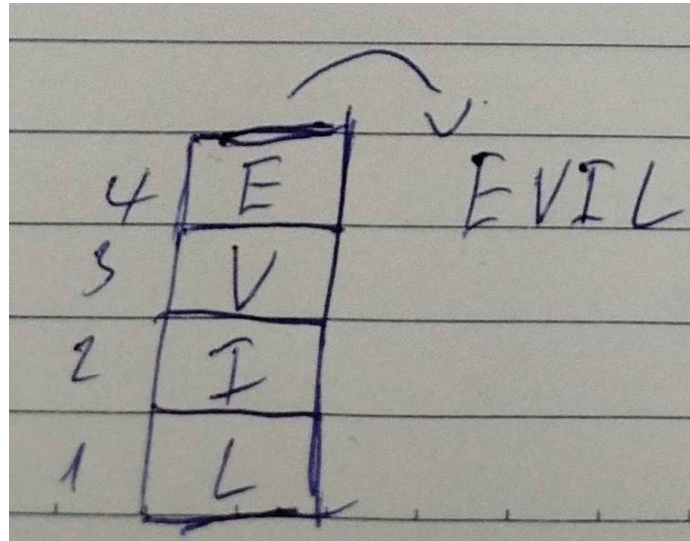


Figure 6: Input: LIVE -> EVIL

Memory Stack.

This section discusses how a memory stack is used to implement function calls in a computer. A memory stack is a designated area of computer memory that holds temporary variables created by functions. When a function is called, its variables are pushed onto the stack. When the function finishes executing, its variables are popped off the stack. This ensures that the memory used by the function is freed up for other purposes. Following Marrtin, 2024. These are some examples of some advantages and disadvantages in using stack memory:

Advantages

- **User control over memory allocation and deallocation:** Stack memory allows the programmer to control exactly how memory is allocated and deallocated. This can be useful for certain tasks, such as implementing real-time systems.
- **Not easily corrupted:** Stack memory is less susceptible to corruption than other types of memory, such as heap memory. This is because stack memory grows and shrinks in a predictable manner.
- **Automatic cleanup:** When a function finish executing, its variables are automatically removed from the stack. This helps to prevent memory leaks.

Disadvantages

- **Limited size:** Stack memory is typically much smaller than heap memory. This means that there is a limit to the amount of data that can be stored on the stack.

- **Risk of stack overflow:** If too much data is placed on the stack, it can overflow, causing a program to crash.

Example: Implementing a Summation Function

The passage concludes with an example of how a stack can be used to implement a function that calculates the sum of numbers from 1 to n . This function works by repeatedly pushing the next number onto the stack and then adding the top two numbers on the stack together. The process continues until all of the numbers have been added.

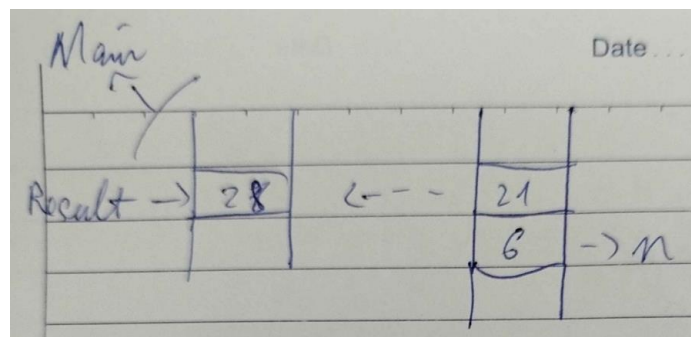
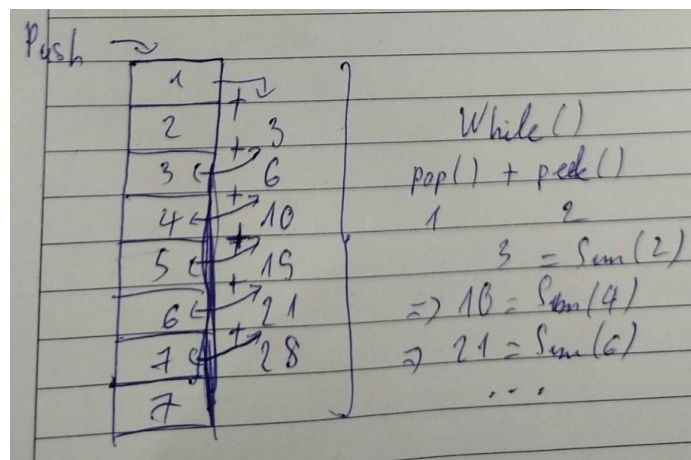
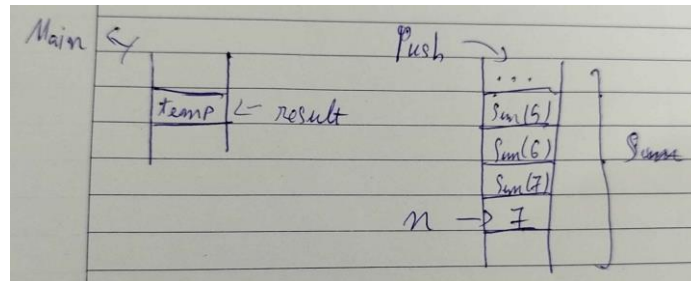


Figure 7 - 8 - 9: Memory stack operation.

By using recursive method, the computer will calculate $\text{Sum}(1)$ then $\text{Sum}(2)$ and go on. Therefore, $\text{Sum}(2)$ is sum of 1 and 2; and $\text{Sum}(3)$ is $\text{Sum}(2) + 3$. The $\text{sum}(7)$ method will return the result = 28.

Queue ADT.

Queue data structures are similar to stacks, but instead of following a Last-In-First-Out (LIFO) order, they follow a First-In-First-Out (FIFO) order. Each element in the queue has two parts: a pointer to the next element in line and a pointer that can hold any type of data (GeeksforGeeks, 2024).

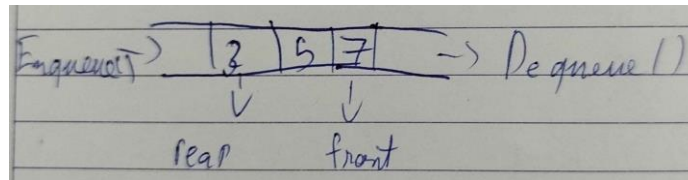


Figure 10: Queue.

The passage mistakenly describes operations for a queue (FIFO - First-In-First-Out) while referencing a stack (LIFO - Last-In-First-Out). Here's a clarification of typical stack operations:

Here's a paraphrased version of the queue operations:

- **add()**: Attempts to insert the given element into the queue if possible.
- **element()**: Retrieves the head of the queue without removing it.
- **offer()**: Tries to add the specified element to the queue.
- **peek()**: Fetches the head of the queue without removing it, returning null if the queue is empty.
- **poll()**: Retrieves and removes the head of the queue, returning null if the queue is empty.
- **remove()**: Retrieves and removes the head of the queue.

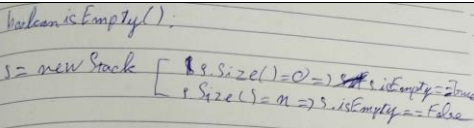
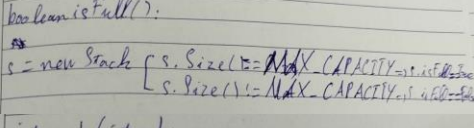
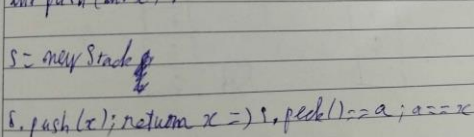
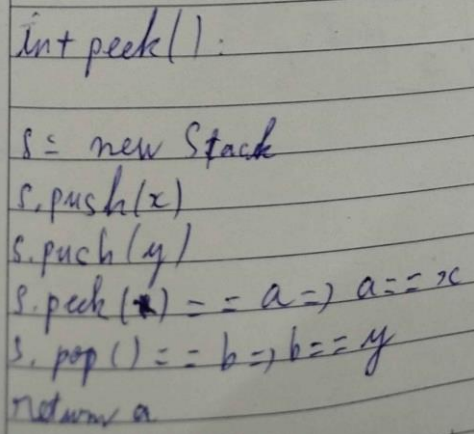
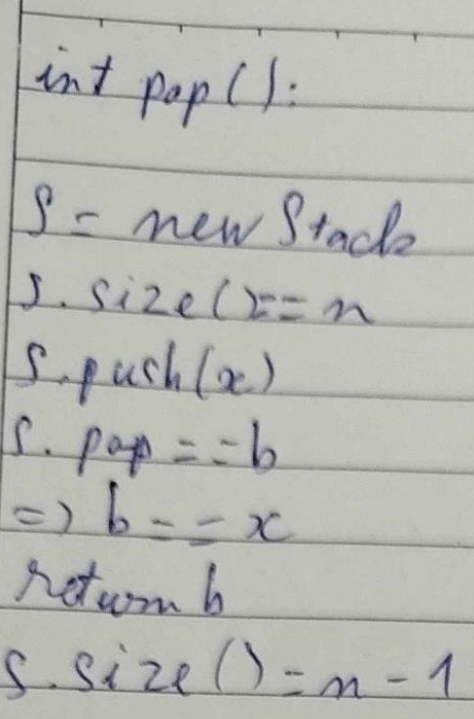
Explanation on how to specify an abstract data type using the example of software stack.

Abstract Data Type (ADT): An ADT is a high-level concept describing how data and operations interact without detailing their implementation.

Stack: A stack is a data structure that follows the Last-In-First-Out (LIFO) principle, meaning the most recently added item is the first to be removed.

- **Stack s**: Represent stack instant.
- **MAX_CAPACITY**: The maximum number of elements the stack can hold.
- **push(x), push(y)**: Adds a new element x or y to the top of the stack.
- **pop()**: Removes the top element from the stack.
- **peek()**: Retrieves the top element of the stack without removing it.
- **size()**: Returns the current number of elements in the stack.
- **a, b**: Variables used in pop() and peek().
- **x**: Parameter used in push().
- **isEmpty()**: Checks if the stack is empty.
- **isFull()**: Checks if the stack is full.

This is the table to explain clearly about operations:

Operation	Precondition	Postcondition	Error Exception
boolean isEmpty()	None		None
boolean isFull()	None		None
int push(int x)	isFull() == False		isFull() == True
int peek()	s.size() > 0		isEmpty() == True
int pop()	s.size() > 0		isEmpty() == True

Summary of the improvements:

- **Clearer postconditions:** The descriptions of what happens after each operation is performed are now more detailed. For example, the pop() operation now explicitly states that it removes the top element from the stack and returns it.
- **Defined error conditions:** The conditions under which an error occurs are now specified. For example, trying to push an item onto a full stack will raise an OverflowError, and trying to pop or peek from an empty stack will raise an IndexError.
- **Completeness:** The table now includes all of the important operations for a stack ADT (push, pop, peek, isEmpty, and isFull), along with their preconditions, postconditions, and error conditions.

References.

Axiomatic and Algebraic Specification in Software Engineering (2023) Webeduclick.com. Available at: <https://webeduclick.com/axiomatic-and-algebraic-specification/> (Accessed: 27 June 2024).

Matthew Martin and Martin, M. (2024) Stack vs Heap Memory – Difference Between Them, Guru99. Available at: <https://www.guru99.com/stack-vs-heap.html> (Accessed: 27 June 2024).

GeeksforGeeks (2023) Abstract Data Types, GeeksforGeeks. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/abstract-data-types/> (Accessed: 27 June 2024).

Java Platform, Standard Edition Java API Reference (2022) Stack (Java SE 18 & JDK 18). Available at: <https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/util/Stack.html> (Accessed: 27 June 2024).

What is Stack Data Structure? A Complete Tutorial (2024) GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/> (Accessed: 27 June 2024).

Bolton, D. (2019) What Is Encapsulation in C++ and C#?, ThoughtCo. ThoughtCo. Available at: <https://www.thoughtco.com/definition-of-encapsulation-958068> (Accessed: 27 June 2024).

Index of comments

2.1 - P1 & P2: The operations of Stack and Queue ADTs have been introduced. The work then describes how the memory stack is used in function calls. However, this description is relatively brief, and the formatting of the figures could be improved.

- P3: The specification for Stack operations is acceptable in content. However, it requires improved formatting for better clarity and presentation.

---***---

Assessor Signature: Hong-Quan Do

Grade: P