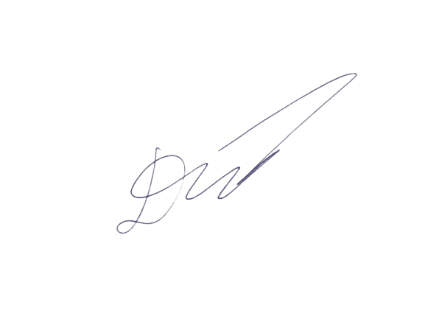
**ASSIGNMENT 1 FRONT SHEET**



|  |  |  |  |
| --- | --- | --- | --- |
| **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** | Mai The Duc | **Student ID** | GCH200681 |
| **Class** | GCH0907 | **Assessor name** | Hong-Quan Do |
| **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:** | | |
| **IV Signature:** | | |

Table of Contents

[Introduction 5](#_Toc112046865)

[I. Abstract data type (ADT) 5](#_Toc112046866)

[I.1. Definition 5](#_Toc112046867)

[I.2. Stack ADT 6](#_Toc112046868)

[I.3. Memory Stack 9](#_Toc112046869)

No table of figures entries found.

# Introduction

I am a software developer working for Softnet Development Ltd. The company providing network provisioning software solution. Softnet has won a value contract last session. We design and develop a middleware solution which can be interface at the front-end to multiple computer provisioning interfaces and the back-end telecom provisioning network.

In my report, I will inform my team about designing and implementing abstract data types. Also, I’ll present how improve software design, development and testing by using ADTs. Furthermore, how to specify abstract data types and algorithms will be introduced.

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

# Abstract data type (ADT)

## I.1. Definition

ADT defines a particular data structure in terms of data and operations. It also offers and interface of the objects as instances of an ADT.

An Abstract Data Type (ADT) consist of:

* Declaration of data.
* Declaration of operations
* Encapsulation of data and operations.

ADT is unclear what algorithms will be used to carry out the operations and how the data will be organised in memory. Because it provides an implementation-independent view, it is called "abstract" (Chauhan, 2022).

Encapsulation means the data is hidden from user and can be controlled only by means of operations. We do not need to know how that data type is implemented, we only need to know how that data type can do. ADT have operations to help us use a data type. I will define Stack ADT and Queue ADT.

## I.2. Stack ADT

Stack is a linear data structure which keeps the operations performed in a specific order (GeeksforGeeks, 2022). Stack is the implantation order of LIFO (Last In First Out) or some call it FILO (First In Last Out).

According to Oracle (2022), Primitive operations of a Stack are:

* empty() : To test if the stack is empty.
* peek() : To looks at the object at the top of this Stack without removing it from the stack.
* pop() : To removes the object at the top of this Stack.
* push(E item) : To pushes an item onto the top of this stack.
* search(Object o) : To search a item in this Stack

+) push

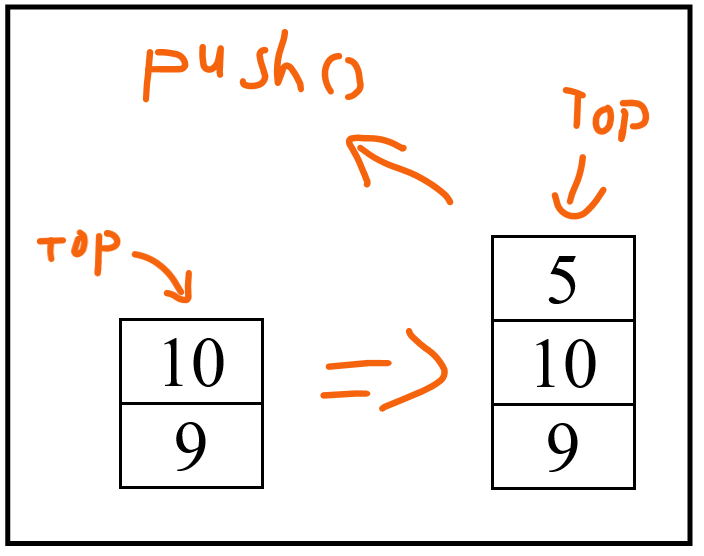


Figure : Stack operation - push()

+) pop

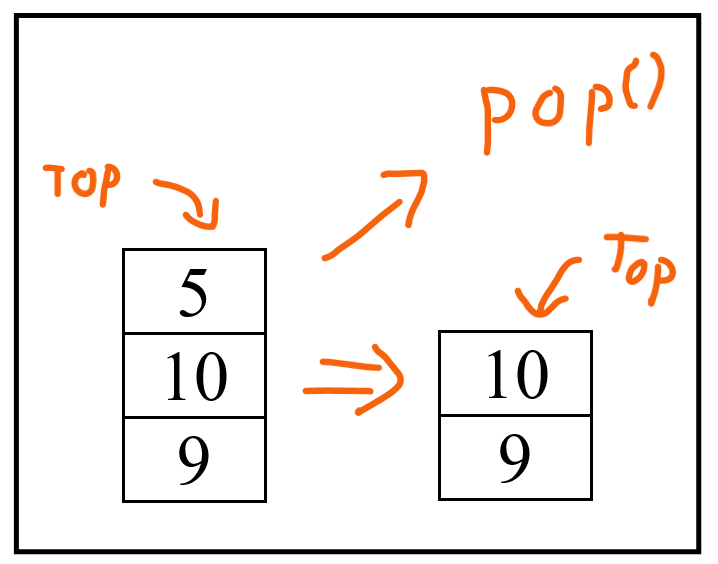


Figure : Stack operation - pop()

+) peek

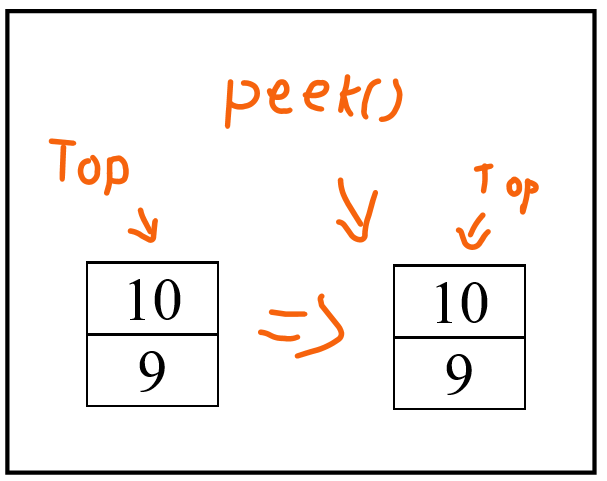


Figure : Stack operation - peek()

Application / example:

A String can be reversed using Stack by adding(push) each character one at a time to the stack and removing(pop) them one at a time. The String will now be returned in the reverse way.

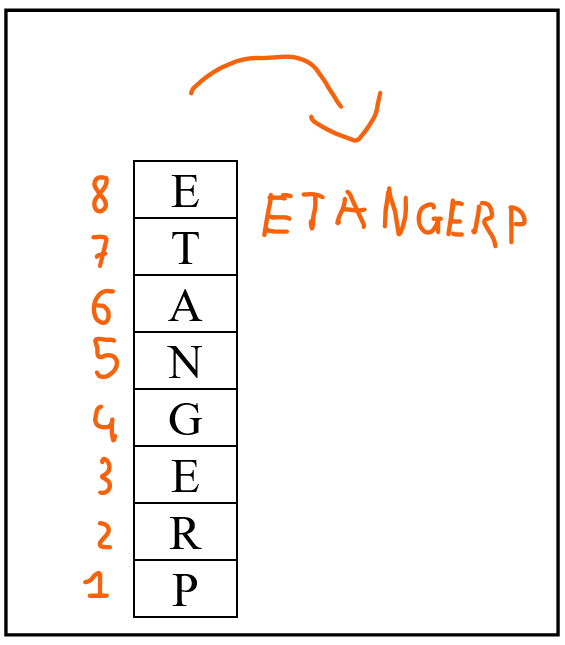


Figure : Stack application/example

Input: PREGNATE 🡪 ETANGERP

## I.3. Memory Stack

A stack is a distinctive section of computer memory where temporary variables created by a function are kept. Variables in a stack are declared, saved, and initialised at runtime.

Accorsing to Martin (2022), Stack have some advantages and disadvantages of using Stack memory.

+) Advantages of Stack memory:

* Allow user to control how memory is allocated and deallocated.
* Stack automatically cleans up the object.
* Not easily corrupted.
* Variables can not be resized.

+) Disadvantages of Stack memory:

* Stack memory is very limited.
* Creating too many object on the stack can increase the risk of stack overflow.

REF