


## ASSIGNMENT 1 FRONT SHEET

<b>Qualification</b>	<b>BTEC Level 5 HND Diploma in Computing</b>		
<b>Unit number and title</b>	Unit 19: Data Structures and Algorithms		
<b>Submission date</b>	7/3/2021	<b>Date Received 1st submission</b>	
<b>Re-submission Date</b>		<b>Date Received 2nd submission</b>	
<b>Student Name</b>	Nguyen Gia Nam	<b>Student ID</b>	GCH190769
<b>Class</b>	GCH0805	<b>Assessor name</b>	Do Hong Quan
<b>Student declaration</b> 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.			
		<b>Student's signature</b>	

### 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:

I. Introduce .....	5
II. A design specification for data structures explaining the valid operations .....	5
1. ADT (P1).....	5
2. Stack ADT .....	6
3. Stack Memory .....	9
4. Queue ADT .....	11
III. The way to specify an abstract data type using the example of software stack.....	14
1. Top (S: Stack, i, r: Item).....	14
2. Push (S: Stack, i: Item).....	14
3. Pop (S: Stack; i, r: Item).....	14
4. isFull (S: Stack, r: bool) .....	14
5. isEmpty (S: Stack, r: bool) .....	14
IV. References .....	15

Figure 1. ADT (Source: studytonight.com, 2021).....	6
Figure 2. An Example of Stack (Source: codelearn.io, 2021).....	7
Figure 3. Peek in Stack .....	8
Figure 4. Stack operations (Source: Tutorialspoint, 2021).....	8
Figure 5. Example of Recursion .....	10
Figure 6. An example of Queue (Source: Tutorialspoint, 2021).....	11
Figure 7. Queue Operations (Source: Studytonight, 2021) .....	13

## I. Introduce

A data structure is a pre-programmed method of storing data for further use. About any business program makes use of a variety of data structures in any way. This guide will provide you with a thorough understanding of Data Structures, which is needed to comprehend the complexities of enterprise-grade applications as well as the requirements for algorithms and data structures.

Data structure is a method of storing and organizing data that ensures its optimal usage. The data model is responsible for two tasks. Then it's structurally complex enough to reflect real-world data relationships. If there is a data method, the structure should be easy to run.

An algorithm is a step-by-step process that specifies a series of instructions to be followed in order to produce the desired result. Algorithms are usually written in several languages, regardless of the base language, allowing an algorithm to be applied in several languages.

Using the software stack as an example, I will create some stacks, queues, and how to describe the abstract data form in this assignment.

## II. A design specification for data structures explaining the valid operations

### 1. ADT (P1)

Abstract Data type (ADT) is an object (or class) for which a set of values and a set of operations are specified.

ADT only defines the operations to be carried out but not how these operations to be carried out. It does not say how memory data are organized and which algorithms to implement the operations are used. It's referred to as "abstract" owing to the abstract view.

Abstraction is known as the process of providing only the necessary and hiding information (GeeksforGeeks ,2017).

## Internal Storage of Abstract Data Type in Java

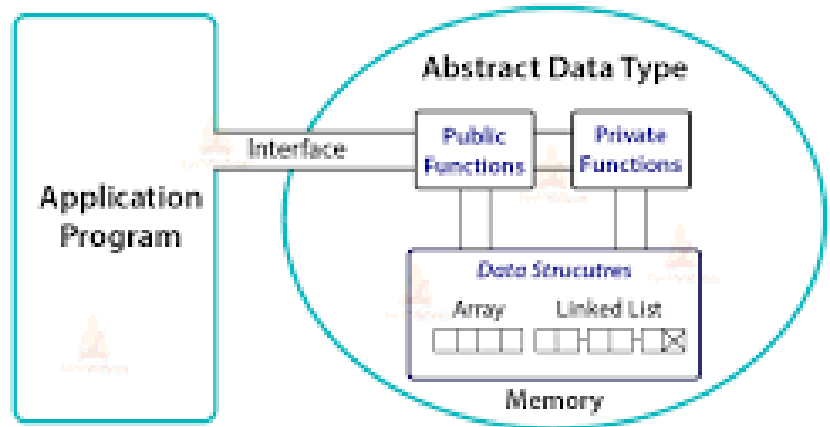


Figure 1. ADT (Source: studytonight.com, 2021)

For example, we used basic values such as int, float, and char data types only with the knowledge that these data types are operated on and performed on without any idea how they are applied. The user needs to know how to implement this type of information. A user also just wants to know what a category of data can do, but not how it is applied. Think ADT as a black box that hides the internal architecture and layout of the records (GeeksforGeeks ,2017).

### 2. Stack ADT

Stack is an abstract data form that acts as a list of items, with two primary operations: Push (adds an element to the stack) and Pop (removes an element from the collection) (removes the most recently added element that was not yet removed). LIFO refers to the order in which elements are removed from a stack (Last In First Out).

A stack can be implemented by means of Array, Structure, Pointer, and Linked List. Stack may be either a fixed size or a complex resizing sensation. Here, we'll implement stack through arrays that make it an implementation of a fixed size stack (Tutorialspoint, 2021).

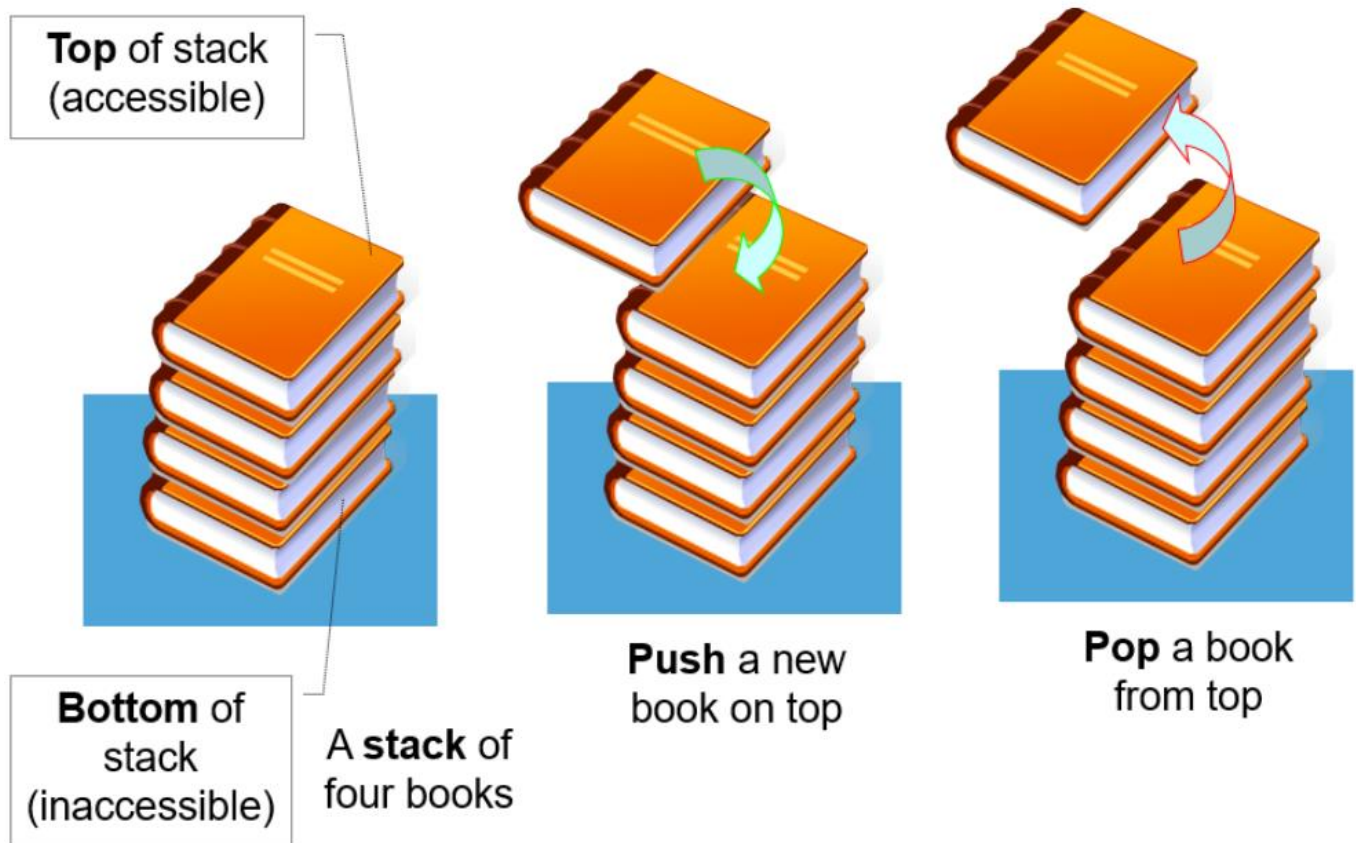


Figure 2. An Example of Stack (Source: [codelearn.io](https://codelearn.io), 2021)

- Basic Operations

Stack operations may involve the initialization, use and subsequent disposal of the stack. A stack is used for the two main operations, aside from these basic elements.

- **push()** – Pushing (storing) an element on the stack.
- **pop()** – Removing (accessing) an element from the stack.

When the data have been PUSHed onto stack.

We must also verify the status of the stack in order to use the stack effectively. The following features are applied to Stacks for the same purpose.

- **peek()** – get the top data element of the stack, without removing it.
- **isFull()** – check if stack is full.
- **isEmpty()** – check if stack is empty.

For all times, we maintain a pointer to the last PUSHed data on the stack. Since this point is still the highest of the stack, hence the top. The top pointer gives the stack the top value without deleting it.

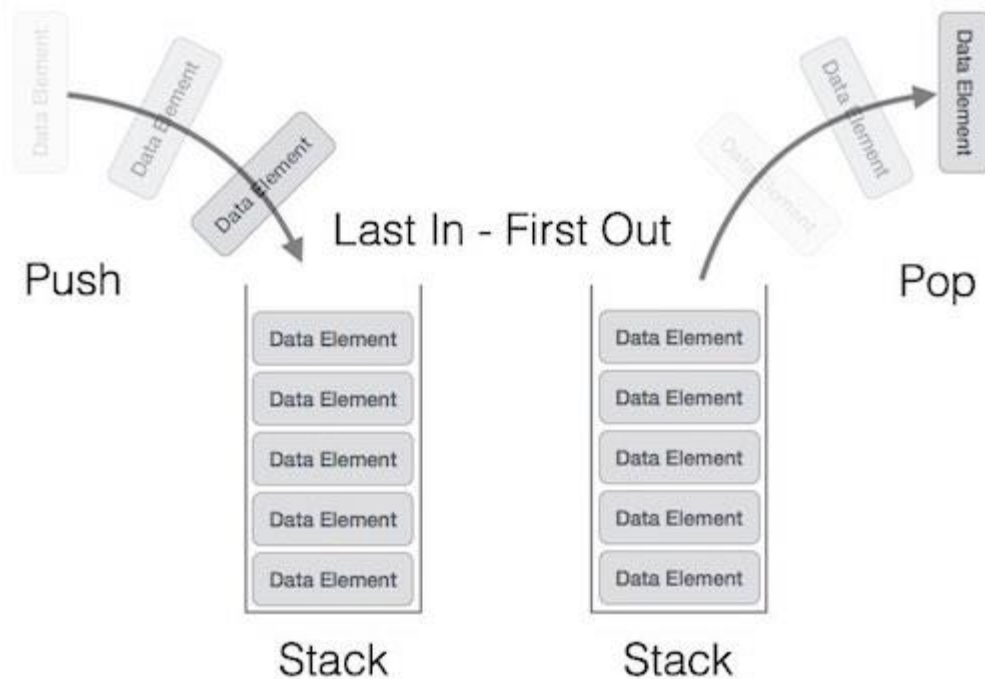


Figure 4. Stack operations (Source: Tutorialspoint, 2021)

## Peek

- **Peek** means retrieve the top of the stack without removing it

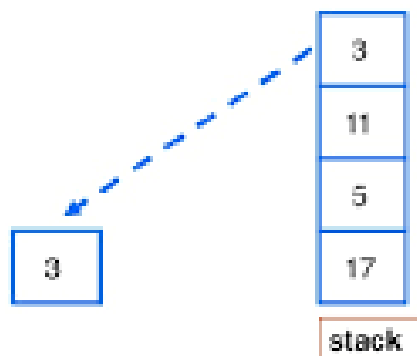


Figure 3. Peek in Stack



- Applications of Stack:
  - “Expression Evaluation. Stack is used to evaluate prefix, postfix and infix expressions.
  - Expression Conversion. An expression can be represented in prefix, postfix or infix notation. ...
  - Syntax Parsing. ...
  - Backtracking. ...
  - Parenthesis Checking. ...
  - Function Call.” (The Crazy Programmer, 2016)

### 3. Stack Memory

Stack memory is used for the execution of each thread.

Stack memory consists of the specific values of the method: local variables and references to objects contained in the heap memory referenced by the method.

Stack memory is referenced in order of LIFO (Last in First Out - at the end, the first output). That is to store the stack type. When a method is executed, a block is created in the stack memory to hold local primitive variables and references to objects. When the method ends, that block will no longer be used and be served on the next method. (STACKJAVA, 2018).

- What is Recursion

Recursion is a computer programming technique that entails calling a method, subroutine, function, or algorithm one or more times before a certain condition is met, at which point the program achieves the desired outcome.

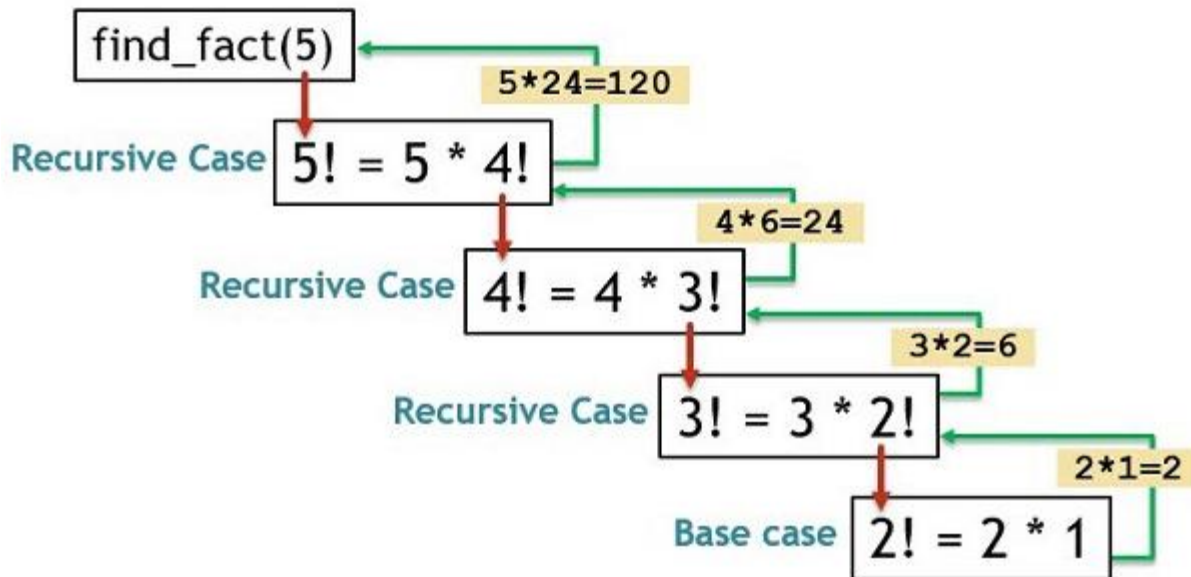
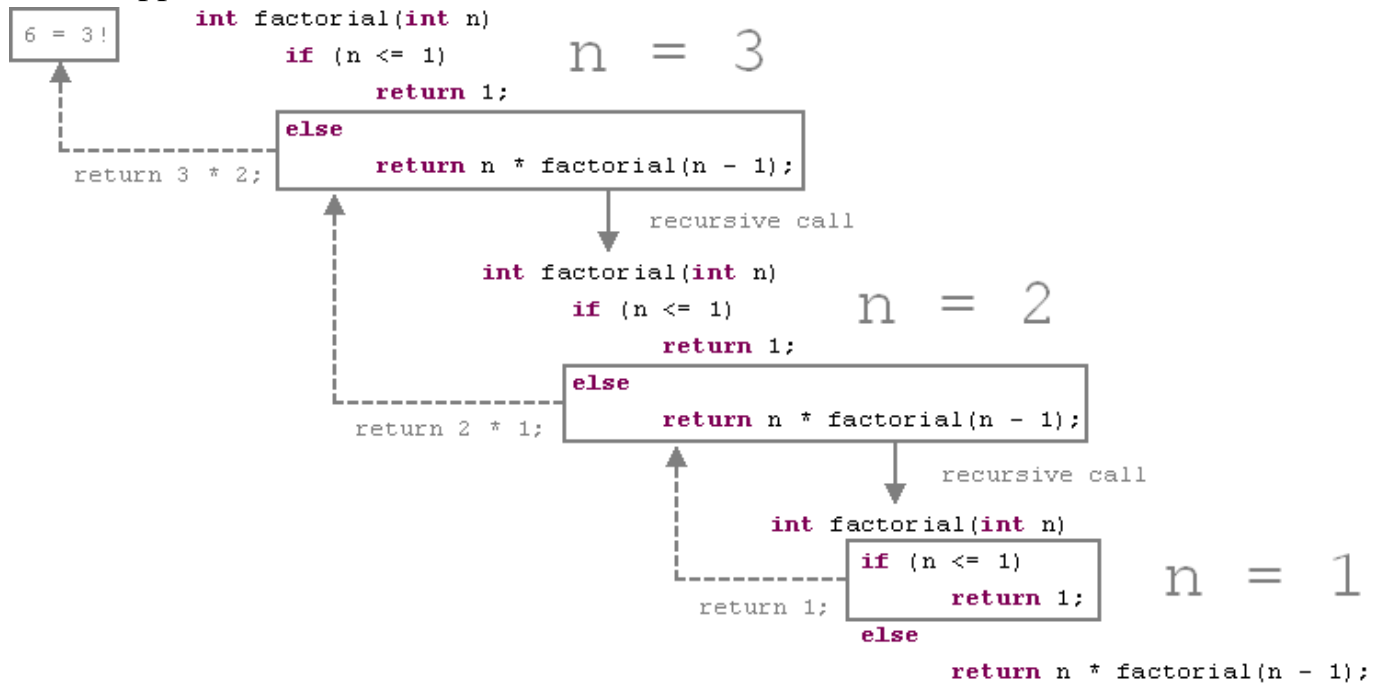


Figure 5. Example of Recursion

- How it is used to implement function calls in a computer

Recursion is the operation of a system repeatedly calling itself. In Java, a recursive process is one that calls itself. It renders the code small but difficult to comprehend. The values passed as actual parameters must be initialized when a function is called. As a result, after the procedure is completed, the system can resume program execution. Methods can be used by other methods or through the main program. We must have certain conditions within the system to avoid the recursive call. Otherwise, the process would be referred to as infinite. To end the recursive call within the process, we use an if... else declaration (or a

similar approach).



#### 4. Queue ADT

Queue is a data structure that is similar to Stacks in that it is an abstract data structure. A queue, unlike a stack, has both ends open. The one end is often used to insert data (enqueue), while the other is always used to remove data (dequeue). The First-In-First-Out (FIFO) approach is used in Queue, which means that the data object that was stored first will be reached first. It using in our daily life (Tutorialspoint, 2021).



Figure 6. An example of Queue (Source: Tutorialspoint, 2021)

- Basic Operations

Initializing or defining the queue, using it, and then completely erasing it from memory are instances of queue operations. Here, we'll focus at the basic operations associated with queues:

- **enqueue()** – add (store) an item to the queue.
- **dequeue()** – remove (access) an item from the queue.
- **front()** Get the front item from queue.
- **rear()** Get the last item from queue.

And Just like the stack, the queue can also use **isFull()** and **isEmpty()**. (Tutorialspoint, 2021)

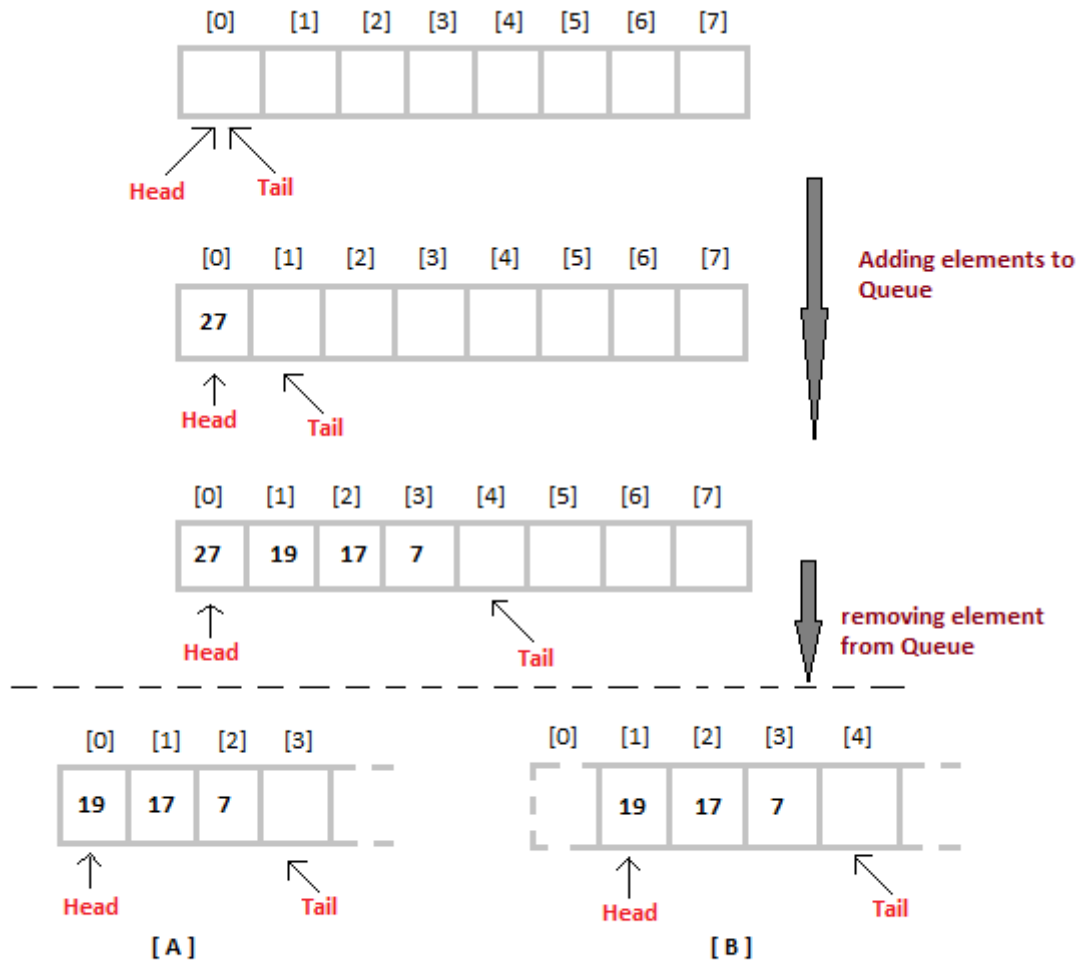


Figure 7. Queue Operations (Source: Studytonight, 2021)

- Applications of Queue

A queue is used if we need to manage a set of items in such a form that the first one in gets out first while the others wait in line, as in the following scenarios:

- Requests are served from a single shared resource, such as a printer, and CPU tasks are scheduled, among other things.
- In the real world, Call Center phone networks use Queues to keep customers who call in line until a service representative is available.
- Interrupt handling of real-time applications. Interrupts are dealt with in the order in which they are sent, i.e., first come, first served (Studytonight, 2021).

### III. The way to specify an abstract data type using the example of software stack.

#### Stack ADT

##### 1. Top (S: Stack, i, r: Item)

- Pre-condition: S is not full
- Post-condition:  $r = i$  after  $i$ (top of Stack) has been popped off Stack.
- Error: Stack is empty  $\rightarrow$  Message: Stack underflow.

##### 2. Push (S: Stack, i: Item)

- Pre-condition: S is not full.
- Post-condition:  $\text{top}(S) = i$  and size of  $S = n + 1$ .
- Error: Stack is full  $\rightarrow$  Message: Stack overflow.

##### 3. Pop (S: Stack; i, r: Item)

- Pre-condition: S is not empty and  $\text{top}(S) = i$ .
- Post-condition:  $\text{top}(S) = r$  and size of  $S = n - 1$ .
- Error: Stack is empty  $\rightarrow$  Message: Stack underflow.

##### 4. ~~isFull (S: Stack, r: bool)~~

- Pre-condition: S is full.
- Post-condition:  $r = (\text{size}(S) = \text{Max\_size})$ .

##### 5. ~~isEmpty (S: Stack, r: bool)~~

- Pre-condition: S is empty.
- Post-condition:  $r = (\text{size}(S) = 0)$  (The open University, 2021).

## IV. References

Abstract Data Types - GeeksforGeeks (2017).

Available at: <https://www.geeksforgeeks.org/abstract-data-types/> (Accessed: 3 March 2021).

Data Structure and Algorithms - Stack - Tutorialspoint (2021).

Available at:

[https://www.tutorialspoint.com/data\\_structures\\_algorithms/stack\\_algorithm.htm](https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm)

Applications of Stack - The Crazy Programmer (2016).

Available at: <https://www.thecrazyprogrammer.com/2016/04/applications-of-stack.html>.

STACKJAVA (2018).

Available at: <https://stackjava.com/java/phan-biet-heap-memory-va-stack-memory-trong-java.html>.

Queue Data Structure | Studytonight (2021). Available at:

<https://www.studytonight.com/data-structures/queue-data-structure>.

# Index of comments

---

- 2.1
- P1: Definition of ADT is shown. Stack and Queue ADT are then shortly introduced. More figures are required to support your discussions.
  - P2: Description of memory stack is shown. An example of recursive function is mentioned, however, how the memory stack is used in this case has not been well-explained.
  - P3: A formal language has been used. The description is short, and some pre-conditions are not correct. Besides, it should include the description of error conditions.