ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 19: Data Structures and Algorithms		
Submission date	11/07/2021	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Nguyen Vu Thai	**Student ID**	GCH190530
Class	GCH0803	**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	Thai

Grading grid

P1	P2	P3	M1	M2	M3	D1	D2

❒ **Summative Feedback:**	❒ **Resubmission Feedback:**	
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
IV Signature:		

Contents

I. Introduction

Data structure is a way to store and organize data in an orderly, systematic way so that data can be used effectively. An algorithm is a collection of steps to solve a particular problem.

This report including 2 parts. Create design specifications for data structures that explain valid operations Can be implemented on the structures, defining the operation of the memory stack and how it is used to Perform function calls in the computer, use the command definition, specify an abstract data type for one software stacks.

II. Part 1: Create a design specification for data structures explaining the valid operations that can be carried out on the structures.

1. ADT

In computer science, an abstract data type is a theoretical data type that is largely defined by the operations and work on it and the limitations that apply. Professionals describe an abstract data type as a "mathematical model" for groups of data types, or as a "value with associated operations" that is independent of a particular implementation.

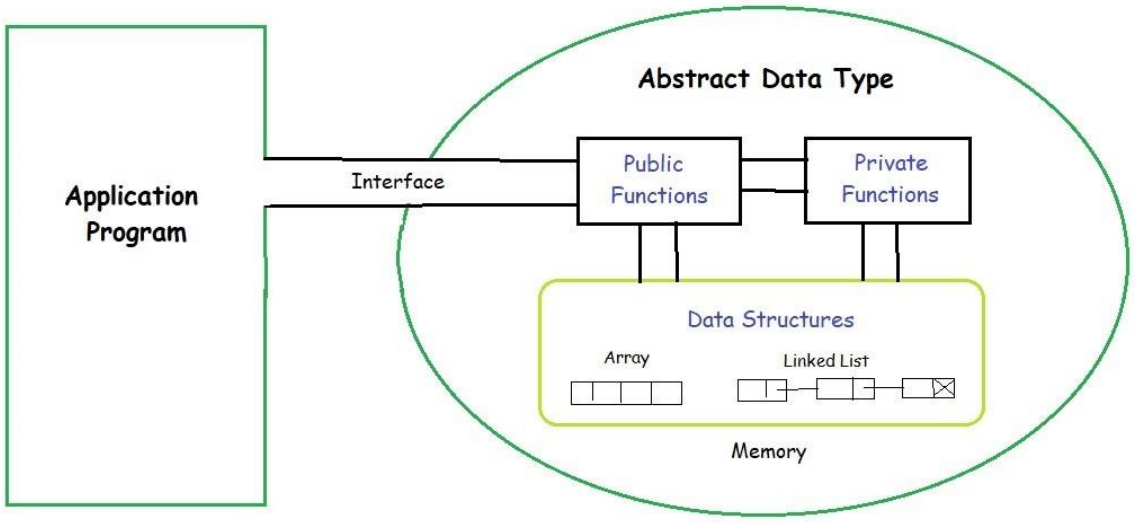

Figure 1: Abstract Data type (According to Geeksforgeeks.org)

When talking about abstract data types, it is helpful to have a knowledge of more concrete data types that are defined more clearly. By contrast, abstract data types are supposed to be broader definitions. For example, an exploration of abstract data types in Java often includes the "List" data type, which can be open to various implementations. IT experts might also use the idea of a "real life" example where a more abstract identifier like "car" represents the abstract data type, and a narrower, more precise identifier like "Nissan Altima" represents a defined or concrete data type.

2. Stack ADT

a. What is Stack ADT?

A stack, which is a basic data structure, can be logically thought of as a linear structure represented by a real physical stack or pile, a structure where insertion and deletion of items takes place at one end called top of the stack. The order may be LIFO (Last In First Out) or FILO (First In Last Out)

b. Stack operations and working mechanism

The stack data structure is limited in the same way. As such, our stack manipulation includes only the following actions:

- Push: Adds an element to the top of the stack, the number of items on the stack increases by 1.
- Pop: Removes the first element from the top of the stack, reducing the number of items on the stack by 1.
- Top: Get the value of the first element at the top of the stack, the number of items on the stack does not change.
- IsEmpty: Checks the stack is empty or not. An empty stack is a stack with no elements.

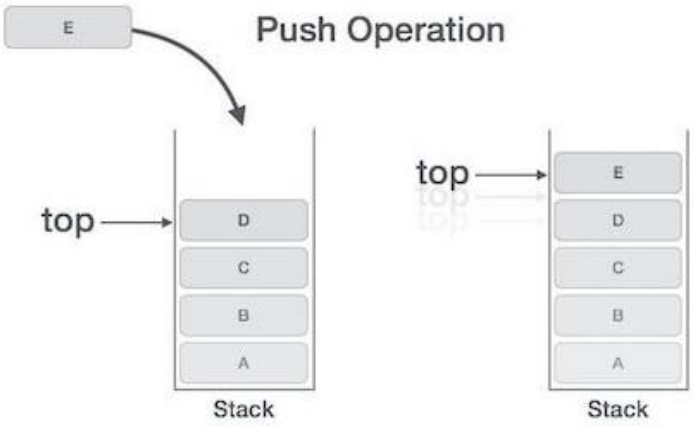

Figure 2: Push Operation (According tutorialspoint.com)

- Step 1: Check the stack is full?
- Step 2: If the stack is full exit program
- Step 3: If the stack is not full, increase top by 1 unit
- Step 4: After increasing, change the top of stack by E (new element have just added)

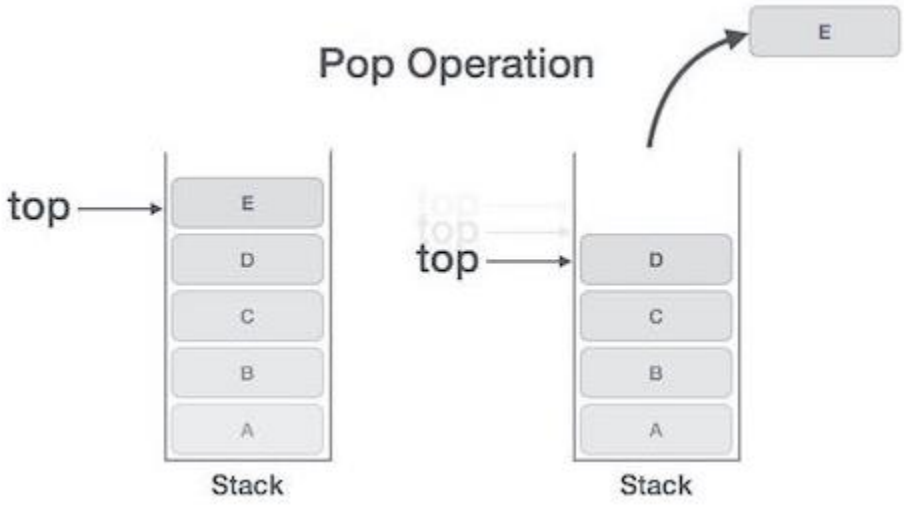

Figure 3:Pop Operation (According tutorialspoint.com)

- Step 1: Check is the stack is empty?
- Step 2: If the stack is empty exit program
- Step 3: If the stack has least 1 element, accesses the data element at top of the stack
- Step 4: Decrease top by 1 unit

c. Stack application

There are many applications for stack. I choose “reverse” String to describe the application of Stack.

Idea: For example, we have 1 string “ABCD”. The first thing, you need to create an array to store each element of the String and a stack to push each element of that String

Go through each element in the chars[] array and use the push() function to put the elements in chars [] into the stack.

Then use the pop () function to convert the top element of the Stack into the char [] array.

Source code:

```
        Stack<Character> stack = new Stack<>();
        String str = new String("ABCD");
        char[] chars = str.toCharArray();
        for(char c: chars) {
            stack.push(c);
        }
        for(int i =0; i<str.length(); i++) {
            chars[i] = stack.pop();
        }
        String s = new String(chars);
        return s;
    }
```

3. Stack Memory

a. Operation of memory stack

Stack memory is used when executing 1 thread of the program. The mechanism of action is LIFO (Last-In-First-Out). When the application starts to execute local variables, the function address, the object reference variable is stored in the Stack blocks, depending on the calling order that the components pushed onto the stack are sorted in the correct order. When a method ends, variable values and relative references are discarded - and the function's address is immediately dropped.

b. How it is used to implement function calls in a computer

To show how the stack memory used in implement function call, I will explain following the example about bellow:

```
2  public class Test {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6          int a = 3;
7          int n = fibo(a);
8      }
9
10     private static int fibo(int n) {
11         // TODO Auto-generated method stub
12         if (n >0) {
13             return n * fibo(n-1);
14         }
15         else {
16             return 1;
17         }
18     }
```

Figure 4

Following the above code, we want to calculate fibo() method with passed valued n =3, so the memory push the fibo(3) on stack , fibo(3) is executing and fibo(3) makes a call to fibo(2) and computer will push fibo(2) to memory and save 3 to slot of fibo(3). Next, fibo(2) again makes a call to fibo(1) and push fibo(1) into stack and save 2 to slot of fibo(2) and we have a fibo(1) =1

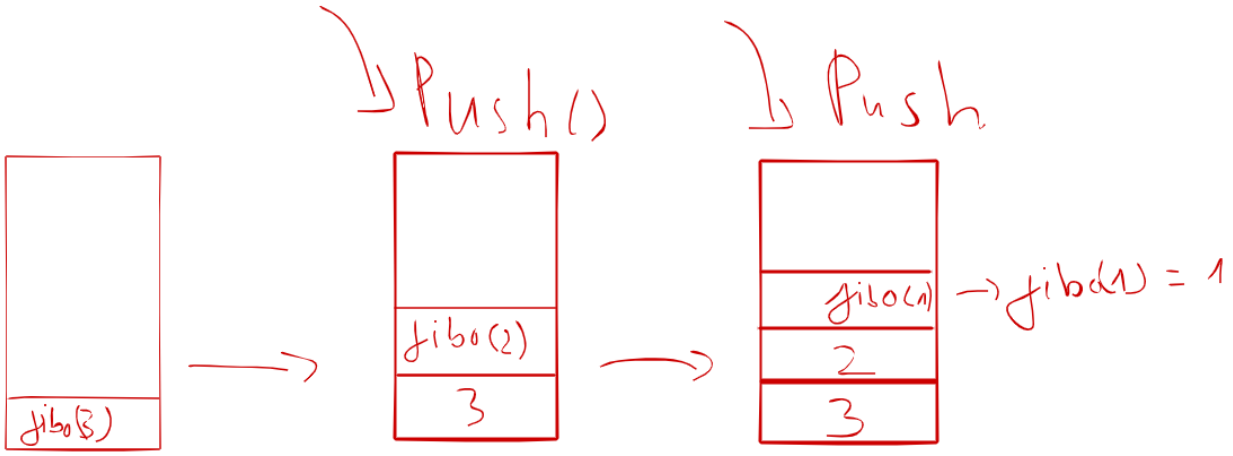

Figure 5

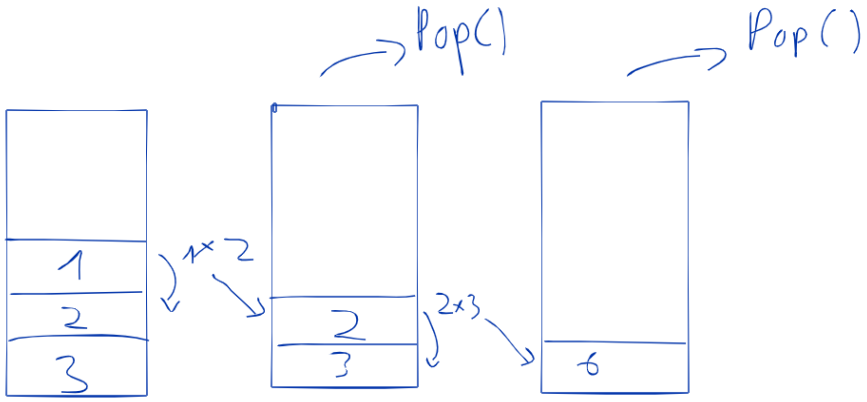

Figure 6

The first, we return 1 as fibo(1) and 1 is popped off from stack. Next return 2 * 1 as fibo(2) and 2 is popped off from stack and return 2*3 as fibo(3) and the function end.

4. Queue ADT

Queue is a data structure used to store objects by FIFO mechanism, meaning "first in first out first".

The image of the queue is very common in everyday life, the image of the queue below is the easiest to understand simulation of the queue data structure: The first person who enters will be greeted first; The newcomer is required to line up at the end.

EnQueue: Adds elements to the last(rear) of the Queue.

DeQueue: Removes the element from first(front) of the Queue. If the Queue is empty, an error message will appear.

IsEmpty: Check that the Queue is empty.

Front: Get the value of the element at the first(front) of the Queue. Getting the value does not change the Queue.

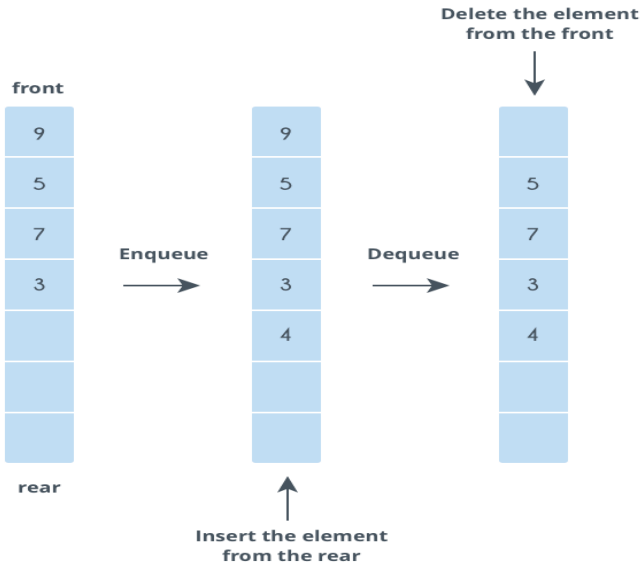

Figure 7: Enqueue and Dequeue operation (According nguyenvanhieu.vn)

Enqueue can be explain:

Step 1: Check if the queue is full, exit program.

Step 2: If the queue is not full, change value of rear by new element that want to add and increase rear by 1 unit.

Dequeue can be explain:

Step 1: Check if the queue is empty, exit program.

Step 2: If the queue can store more element, shift all the elements from index 2 till rear to the right by one

Step 3: Store 0 at rear and decrement rear

III. Part 2: Using an imperative definition, specify the abstract data type for a software stack.

Introduction about formal specification language

The Vienna Development Method (VDM) founded on Hoare Logic supplied a basis for the realization of these concepts in software design but various less formal specification languages are now available. OCL for example, an adjunct to the Unified Modelling Language (UML), allows the use of formal English to define conditions and invariants. (Humby, 2016)

Pre-condition and Post-condition

Some functions will only produce the required result if certain conditions pertain when the function is called. For stand alone functions that do not result in side effects on persistent data, such conditions can only relate to restrictions placed on the input set. For operations on a data structure the current state of the structure may also need to be taken into account. These prerequisites for calling a function are its preconditions.

After the function exits another set of conditions will apply. A statement of these postconditions is needed to check that a particular implementation of the function works according to requirements. (Humby, 2016)

Stack's operations with specification language

Stack: S

- n = size(S): the current number of elements in the Stack
- Y = max length of Stack S

push(int x): void; a = top(s)

- pre-condition: n < Y
- post-condition: a = x; size(S) = n +1
- error-condition: n >= Y: Print "Stack is full"

pop(): Item I; a = top(S)

- pre-condition: n>0
- post-condition: size(S)= n-1; a = I
- error-condition: n <= 0: Print "Stack is empty"

top(S): Item r;

- pre-condition: n>0
- post-condition: r = I {Where I = pop()}
- error-condition: n <= 0: Print "Stack is empty"

size(S): Item r; k is total valid pushes and j is total valid pops since Stack was created)

- pre-condition: k >= j
- post-condition: r = k - j
- error-condition: r < 0: Print “Size(S) can’t be a negative number”

IV. References

GeeksforGeeks. 2021. *Abstract Data Types - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/abstract-data-types/> [Accessed 25 April 2021].

Learn1.open.ac.uk. 2021. *Algorithms 10: Abstract data types – Stack*. [online] Available at: <https://learn1.open.ac.uk/mod/oublog/viewpost.php?post=162582> [Accessed 25 April 2021].