STUDENT'S DECLARATION OF ORIGINALITY

By submitting this online assessment, I declare that this submitted work is free from all forms of plagiarism and for all intents and purposes is my own properly derived work. I understand that I have to bear the consequences if I fail to do so.

Final Online Assessment Submission

Course Code:	**BACS2063**
Course Title:	**DATA STRUCTURES AND ALGORITHM**
Signature:	*JOAN*
Name of Student:	**JOAN HAU**
Student ID:	**20WMR09465**
Programme:	**RSF2G5**
Date:	**1/10/2020**

Question 1	
Question 2	
TOTAL	

QUESTION 1 (a)

(i) The shortcomings of measuring an algorithm's time complexity by finding out the time elapsed between start and end time of running the algorithm are the running times of two algorithms are difficult to directly compare, the comparisons among the running times of two algorithms can only be measure accurately if there are performed in the same hardware and software under the same conditions. Besides, the algorithm must be fully implemented in order to execute it to study its running time experimentally. Furthermore, the algorithm may not perform same time efficiency on different inputs, for example it may be faster in some of the expert hardware or software support compared to low level of hardware or software support. The algorithm may run faster on some inputs than it does on others of the same size.

(ii) The Big-O notation allows us to evaluate the relative efficiency of any two algorithms in a way that is independent of the hardware and software, which means all of the results obtained will not be affected by the hardware or software that is supported. The Big-O notation is performed by studying a high level description of the algorithm without need for implementation and it also takes into account all the possible inputs. Below is the example of code segment :

```
Algorithm
sum = 0
for i = 1 to n
  sum = sum + i
```

	Algorithm
Assignments	n + 1
Additions	n
Multiplications	
Divisions	
Total Operation	2n + 1

Total Operations

= total assignments + total additions

= **(1 assignment for initializing sum + n assignments in for loop body)** + *(n additions in for loop body)*

= **(1 + n)** + *(n)*

= **2n + 1**

Algorithm (Big O of n)

2n + 1

= 2n, ignoring smaller terms

= n, ignoring the coefficient 2

=> O(n)

(iii)

A: 1000 / 10 = 100
2000 / 100 = 20 seconds

B: 1000 ^ 2 = 1000000
2000 ^ 2 = 4000000
4000000 / 100000 = 40 seconds

QUESTION 1 (b)

The Queue ADT can be used when there are first-in, first out (FIFO) principles that are needed for the operation. For example, the customer service system. The customer service staff may need to chat with the customer when there are any customer who face the difficulties, the customer will be arranged in a queue waiting for the staff to serve them, the customer who enter the system or request for help first will be turn to he or she first which means that the FIFO principles is apply to simulate the waiting line. The responsibility of the customer service system will simulate the customer entering and leaving a waiting line in order to arrange the customer order. The object in the queue will be the customer's arrival time, customer's transaction time and customer's number which the details related to the customer. Besides, the wait line will help to simulate the customer entering and leaving the waiting line which may include the line (which contains a queue of customer), number of arrivals (which contains of number of customer), number served (number of customer served by the system or staff), total time waited (the total time waited for the customer to wait for their turn). This ADT will be work in such a way that when there is any customer enter to the system, the customer will be added into the rear of the list, and the customer will be continue added when there is any customer request for serve, then the system will call the customer based on their sequence, this ADT will again dequeue the customer which located at the front of the queue from the waiting line. This ADT will apply the FIFO principles for the whole operation.

QUESTION 1(c)

TWO benefits of using abstraction in the data structures

1. Modularity
   By using the abstraction in data structures, different components of a software system are divided into separate functional units such as function, method, class, package, module, library or subsystem and so on, which can help in developing a software system with strong cohesion and loose coupling and it may help in the operation and maintenance of the system. The abstraction in data structures can help to distill a complicated system down to its most fundamental parts and just focus on what should be done in the specific part with simple, precise language by naming the parts and explaining their functionality.
2. Reliability
   Abstraction means hiding the implementation details by providing 1 layer (interface) over the basic functionality. By hiding the implementation details, user code cannot directly access the objects of the type or depend on the representation, which may allow the representation to be changed without affecting user code. Besides, the

programmer has freedom in implementing the details of the system as all of the details will be hidden and the user will not observe any abstract code from the client side, the programmers only need to maintain the contract (interface) that outsiders see.

QUESTION 1(d)

(i)

```
public static void pattern(int n){
    if(n != 0){
        System.out.print("*");
        pattern(n - 1);
    }

}

public static void drawPattern (int n, int i) {

    if (n != 0){
        pattern(i);
        System.out.println();
        drawPattern(n - 1, i + 1);
    }
}
```

(ii) The recursive solution will be more suitable for this situation. This is because the recursive solution is simpler and easier to understand than an iterative implementation. A recursive approach to algorithms allows us to take advantage of the repetitive structure present in many problems like example folders that have subfolders. Besides, a recursion can make code smaller and lesser while iteration makes it longer and complex to read and used.

QUESTION 2(a)

The suitable Abstract Data Type that is used to store the search object of a book is sorted lists with array implementation. By using the sorted list, we are able to automatically sort the elements entered into the system and this ADT will help us to rearrange the list which will place the list in ascending order and we are able to save our time to manually sort the list in order. As the instruction given that the ADT should be able to be sorted by the title in ascending order (A - Z), and the sorted list has fulfil this requirement as it will automatically sorted the list once an entry appears in the list. Besides, by using the sorted list with array implementation, we can apply the binary search to develop this search function. The binary search can only apply to sorted arrays, therefore this sorted list with array implementation will be able to fulfil this rule. The binary search will work in such a way that repeatedly

divides the array in half until the element is found or there is nothing left to search. This can be used when the user enters a word or book title into the system to search for a book from the database, then the binary search may take place to provide the effective search method for the user. The binary search is a faster method for searching compared to sequential search. The best case for the efficiency of the binary search is O(1) which is very fast in searching and can help in decreasing the time consuming.

QUESTION 2(b)

(i) Insertion sort

(ii) Yes the sorting algorithm is suitable to be used to sort the book search. The insertion sort may only be acceptable for small arrays size which means the maximum array length for the result return is 20, which is considered as small size in array, therefore this sorting is suitable under this condition. Besides, the insertion sort is simple to code and has a very good performance with small lists which can help in sorting the book results and reduce the time consuming for sorting. Moreover, the insertion sort is very good when the list is almost sorted and this will give help for the sorting which means less work the insertion sort does and more efficient the sort is. Furthermore, the insertion sort is quite good with sequential data that is being read in one time, as the system only displays the output based on the user's input data at the time, therefore the data is just being read in one time.

(iii)

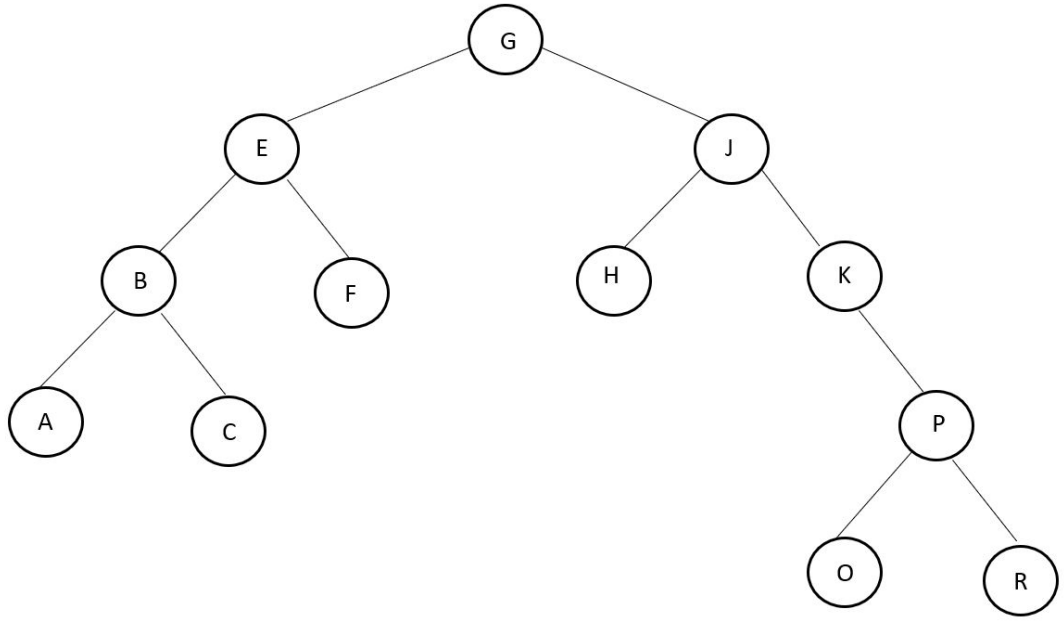
	0	1	2	3
Original	"The Abundance of Less"	"Clutter-Less"	"The Year of Less"	"The Simplicity of Less"
After Pass 1	"Clutter-Less"	"The Abundance of Less"	"The Year of Less"	"The Simplicity of Less"
After Pass 2	"Clutter-Less"	"The Abundance of Less"	"The Year of Less"	"The Simplicity of Less"
After Pass 3	"Clutter-Less"	"The Abundance of Less"	"The Simplicity of Less"	"The Year of Less"

QUESTION 2(c)

(i) Inorder - A,B,C,E,F,G,H,J,K,M,O,P,R
Postorder - A,C,B,F,E,H,K,O,R,P,M,J,G

(ii) To remove the 'M', from the tree, find the largest value from the left subtree of the value 'M' and replace the 'M' with value 'K' by setting the parent of value 'K' to value 'J' and then

remove the M.



QUESTION 2(d)

```
public int getHashIndex(String studID){
        int hashCode = Integer.parseInt(studID.substring(5));
        int hashIndex = hashCode % 83;
        if(hashIndex < 0)
                hashIndex = hashIndex + 83;
        return hashIndex
}
```