Module: CT2109 OOP: Algorithms and Data Structures

Date: 27/05/2021
Student name: Cathal McSweeney
Student ID: 19731485

Student statement of academic integrity: In submitting this work I confirm that it is entirely my own. I acknowledge that I may be invited to online interview if there is any concern in relation to the integrity of my exam.

Disclaimer that you must include at the top of your submission:	In submitting this work, I confirm that it is entirely my own. I acknowledge that I may be invited to online interview if there is any concern in relation to the integrity of my exam, and I am aware that any breach will be subject to the University's Procedures for dealing with breaches of Exam Regulations: https://www.nuigalway.ie/media/registry/exams/QA230---Procedures-for-Dealing-with-Breaches-of-Examination-Regulations.pdf

(21) a)

Big O-notation is the theoretical analysis of a snippet of code using high level pseudo code.

essentially you Add up the lines of code in terms of N which varies depending on the code such as a primitive operation = 1 and a loop would = N with all primitive operations within the loop = N-1

```
Int I = 0                          1
Int B = 6                          1
Int A = 4                          1
For (I; I < A; I++) {              N
    B = B + A;                     N-1
    A = A + I;                     N-1
}
System.out.println(A + " " + B);   1
```

Big O = 3N + 1

b) Fibonacci is a sequence of numbers that are found by adding the two numbers before it together.

First 10 in the sequence

0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

Using a Recursive method to find the N^{th} value of Fibonacci is an effective way of calculating the number. As it calls itself many less lines of code need to be written

Big O of Fibonacci = O(n)

c) Iterative programming is the method of using data in a loop to find a answer through repeating a line of series of lines of code.

Recursive is the method of creating a function that calls itself to complete a function.

```
public int FactorialRecursive (int input){
    if (input = 0){
        Return 1;
    }
    else {
        return input * FactorialRecursive (input-1);
    }
}
```

(i) Theoretical Analysis.

- uses high level pseudocode description of the algorithm. counts the total number of primitive operations in terms of N which is then used to compare algorithms against eachother. Big O is an example of theoretical analysis where each line is given in terms of N with loops and nested loops carrying a heavier N value.

(ii) to carry out an experimental comparison of two different algorithms. For this example we will choose bubble sort and selection sort to compare sorting an array of numbers.

In these two algorithms we will use a counter for both that will increment each time by the standard value applied to each line of code. eg: "i = 0;" will carry a weight of 1 where a loop will carry the weight of 3 with each line within it carrying the weight of 1.

Both algorithms will also be timed to identify which one not only has the higher number of primitive operations but which algorithm takes the longest amount of time.

These two results will form the basis of the comparison which will identify which may be the more efficient algorithm for what's needed.

(iii) The advantages of the two methods would be that the theoretical comparison saves time as it allows one to ~~guess~~ the expected result of the algorithm where as the advantage of ~~the~~ the experimental method allows for a precise result when comparing the effectiveness of two Algorithms.

(a 2) 12, 78, 34, 69, 14, 23, 56, 45, 88

Pivot moved to end

12, 78, 34, 69, 88, 23, 56, 45, 14

Check left bound which moves from left to right until it finds num > pivot.

left Bound = 78

Right bound then moves from Right to left until it finds a value less than pivot or crosses left bound.

Right crosses left bound
pivot & left bound swap.

12, 14, 34, 69, 88, 23, 56, 45, 78

Quick Sort is then called on Right subtist

Pivot = 23

12, 14 34, 69, 88, 23, 56, 45, 78

left Bound = 34
Right Bound moves left until value less than 23 is found
Right bound crosses left bound again
23 & 34 swap

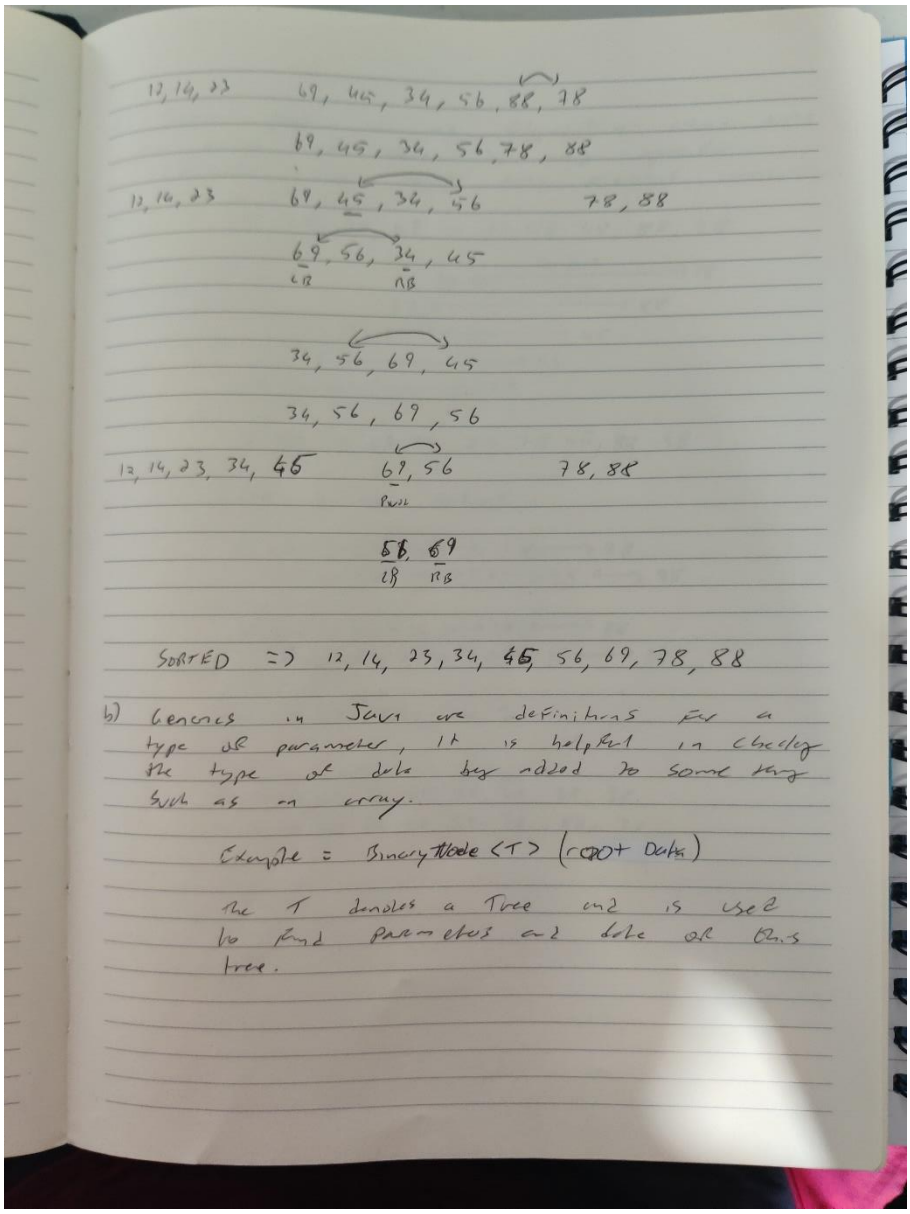
12, 14, 23 69, 88, 78, 56, 45, 34

pivot

69, 88, 34, 56, 45, 78

LB RB

69, 45, 34, 56, 88, 78


12, 14, 23 69, 45, 34, 56, 88, 78

69, 45, 34, 56, 78, 88

12, 14, 23 69, 45, 34, 56 78, 88

69, 56, 34, 45
LB RB

34, 56, 69, 45

34, 56, 69, 56

12, 14, 23, 34, 45 69, 56 78, 88
Pivot

56, 69
LB RB

SORTED => 12, 14, 23, 34, 45, 56, 69, 78, 88

b) Generics in Java are definitions for a type of parameter, it is helpful in checking the type of data being added to some thing such as an array.

Example = BinaryNode <T> (root Data)

The T denotes a Tree and is used to find parameters and data of this tree.

c) Shell sort is an algorithm that sorts an array of data by using specific intervals and sorts each element individually.

12, 78, 34, 69, 14, 23, 56, 45, 88, 98.

14 ⟷ 98 (compare)
69 ⟷ 88
34 ⟷ 45
78 ⟷ 56
12 ⟷ 23

12, 56, 34, 69, 14, 23, 78, 45, 88, 98

Gap is now halved

12 ⟷ 34 ⟷ 14 ⟷ 78 ⟷ 88
56 ⟷ 69 ⟷ 23 ⟷ 45 ⟷ 98

12 ⟷ 14 ⟷ 34 ⟷ 78 ⟷ 88
23 ⟷ 45 ⟷ 56 ⟷ 69 ⟷ 98

Last GAP size = 1

12, 23, 14, 45, 34, 56, 78, 69, 88, 98
12, 14, 23, 34, 45, 56, 69, 78, 88, 98

D) Comparable is where an element is able to be compared itself to another element this allows the code to compare a list of data and compare a single data in them.

Code: ~~Int CompareTo~~

```
Int A = 65
Int B = 78
Int comparison = A.CompareTo(B);
```

Comparator the compares two objects and stores the answer in a new variable

```
Int Compare (Object A, Object B);
```

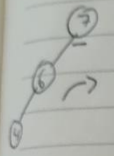
OOP EXAM

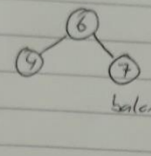
23) a) (i) 7, 4, 6, 22, 29, 23, 3

1) 7 balanced

2) 7 — 4 balanced

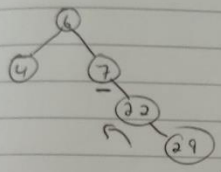
3) 7 — 4 — 6 unbalanced double rotation needed!



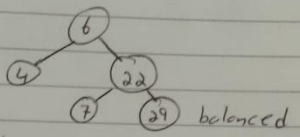


balanced

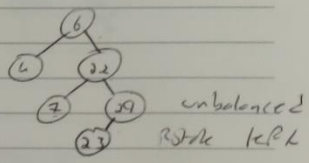
4) 6 — 4, 7 — 22 balanced

5) 

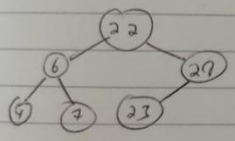
unbalanced left Rotate



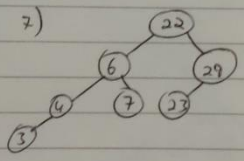
balanced

6) 

unbalanced Rotate left

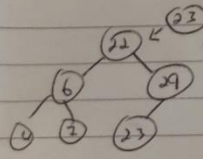


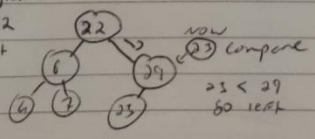
balanced

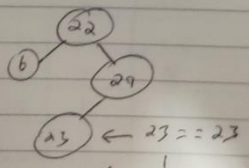
7) 

balanced.

ii) When searching for the number 23 in the tree it compares the number to be searched against the ~~root~~ Root node, if bigger go to right child if small or go left child.



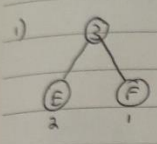




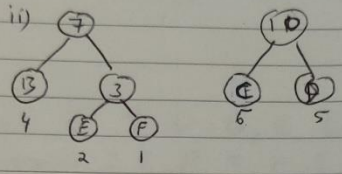
b) Huffman encoding is the encoding of a series of bits by identifying the most frequent bits and having them easier to be accessed in a tree with less frequently used bits being further down the tree.

x	x	x	x	x	x	x
6	4	5	5	2	1	7
A	B	C	D	E	F	G

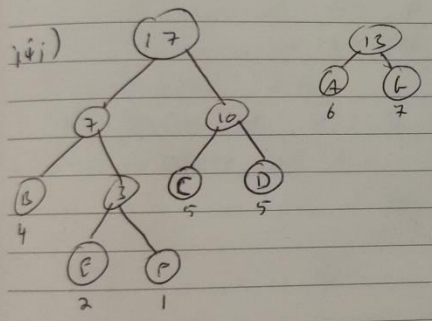
i)



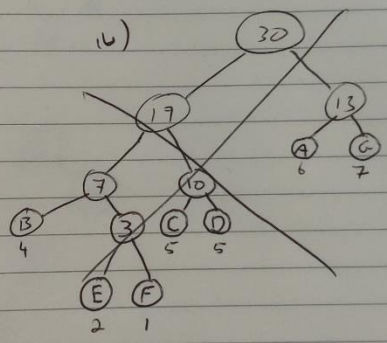
ii)



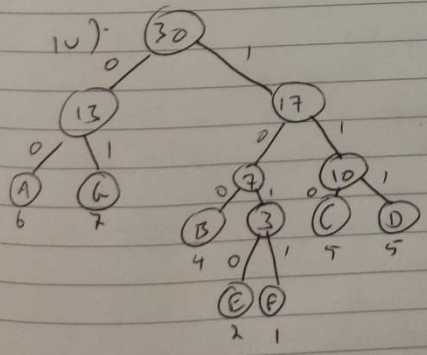
iii)



iv)

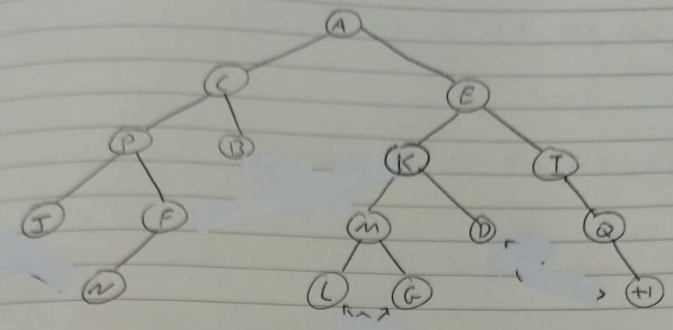


iv):



	Freq	bits
A = 00	6	12
B = 100	4	12
C = 110	5	15
D = 111	5	15
E = 1010	2	8
F = 1011	1	4
G = 01	7	14

c)



i= Leaves = J, N, B, L, G, D, H

ii: Internal nodes = C, P, F, E, I, Q, K, M

iii: Parent of D = K

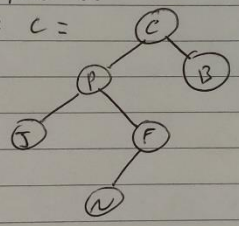
iv: Ancestors of F = P, C, A

v: descendents of I = Q, H

vi: Height = 4

vii: Depth of K = 2

viii: Subtree of C =



D) Compression in Computing is necessary as it allows for quicker computing of data and will allow for faster transmission of data from one place to another ie. internet.

lossy compression = is the reduction of data using inexact and approximations of the data. it is irreversible

Example : JPEG

lossless compression = reduces files to the best size possible without losing the data. Can recover the data easily Example = Huffman coding.