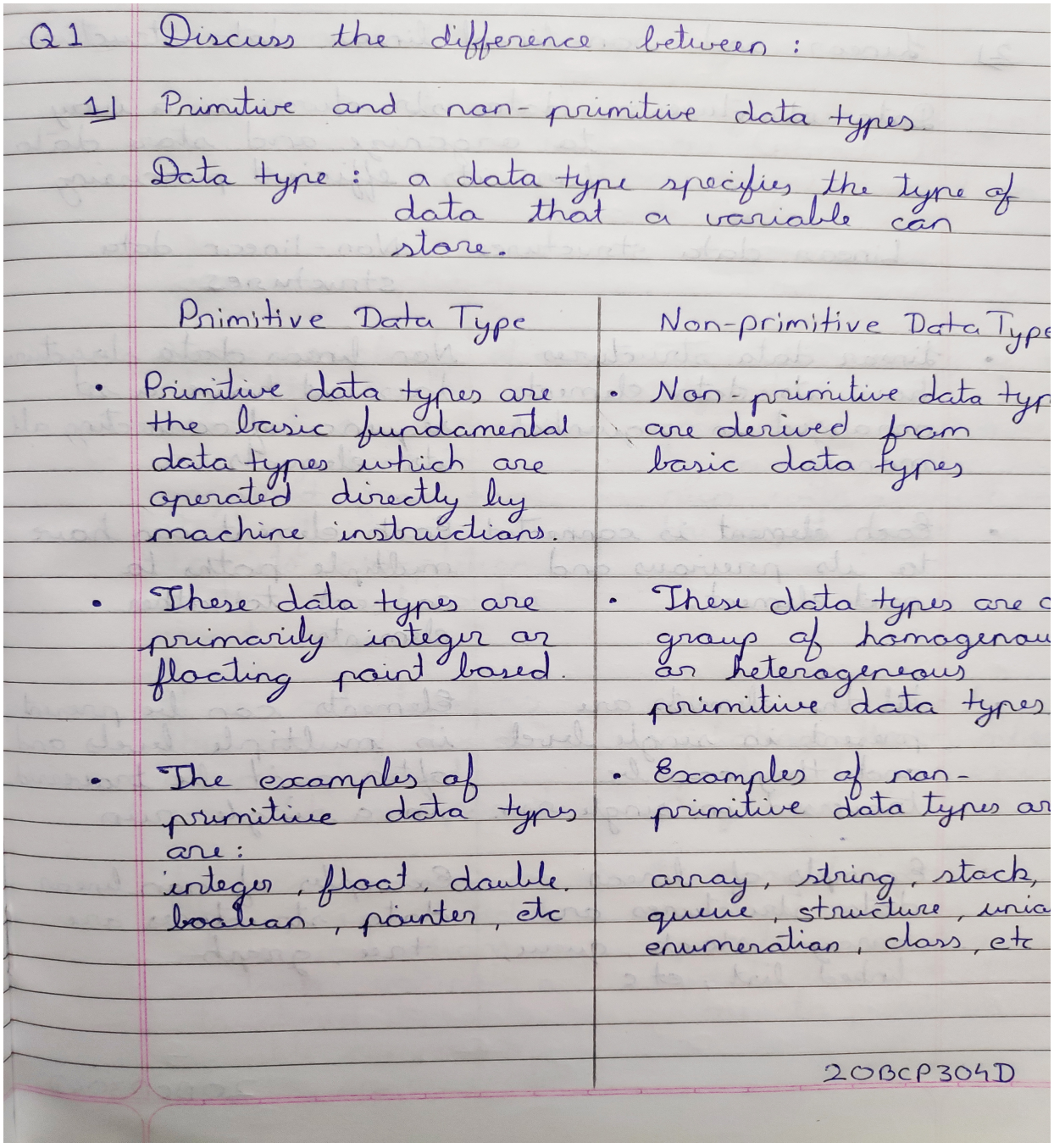
Assignment - 1

Q1. Discuss the difference between:

A. Primitive and Non-primitive data types.



Q1 Discuss the difference between :

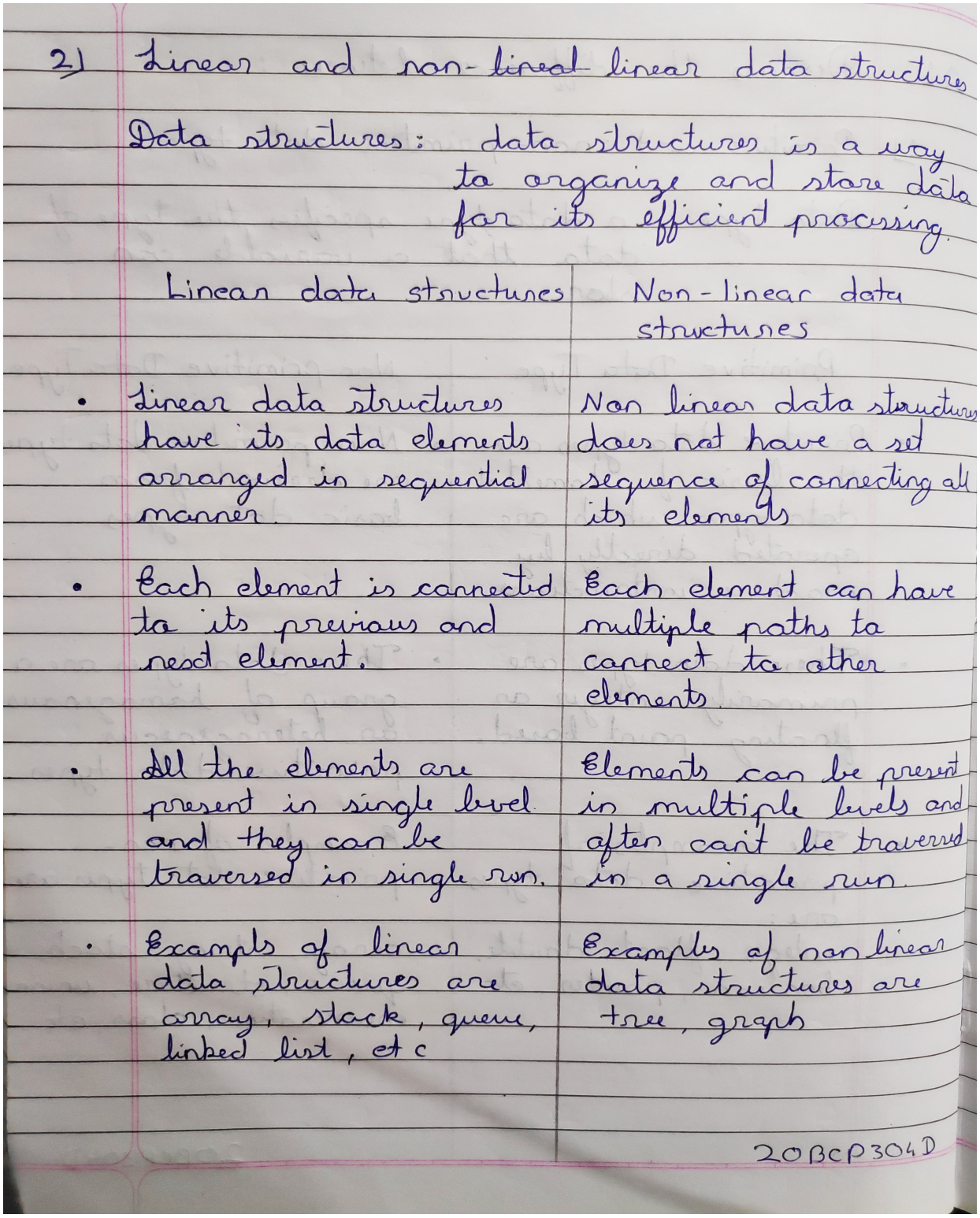
1) Primitive and non-primitive data types.

Data type : a data type specifies the type of data that a variable can store.

Primitive Data Type	Non-primitive Data Type
• Primitive data types are the basic fundamental data types which are operated directly by machine instructions.	• Non-primitive data typ are derived from basic data types
• These data types are primarily integer or floating point based.	• These data types are a group of homogeneou or heterogeneous primitive data types
• The examples of primitive data types are: integer, float, double, boolean, pointer, etc	• Examples of non-primitive data types ar array, string, stack, queue, structure, unio enumeration, class, etc

20BCP304D

B. Linear and Non-linear data structures.



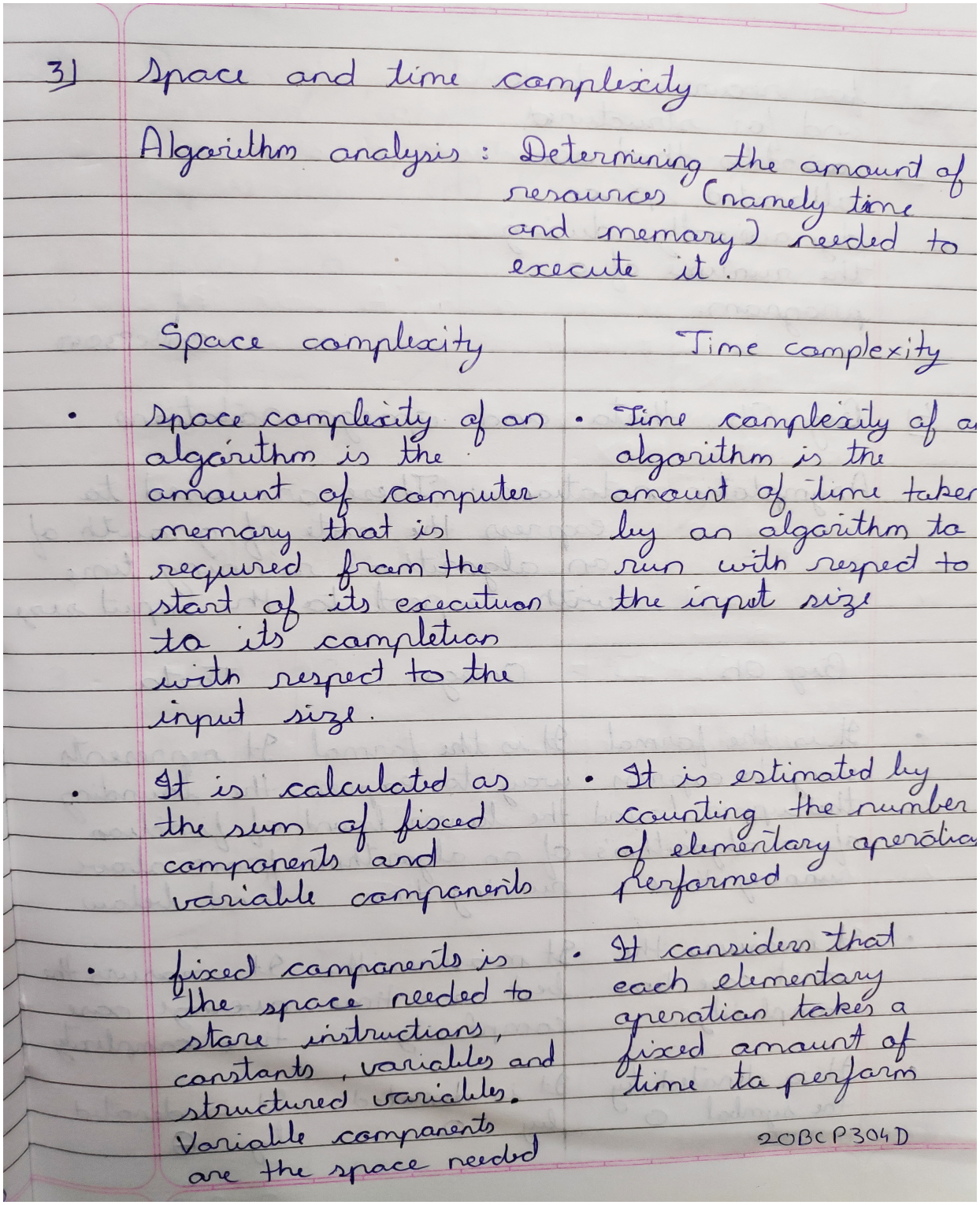
2) Linear and non-linear data structures

Data structures: data structures is a way to organize and store data for its efficient processing.

Linear data structures	Non-linear data structures
• Linear data structures have its data elements arranged in sequential manner.	Non linear data structures does not have a set sequence of connecting all its elements
• Each element is connected to its previous and next element.	Each element can have multiple paths to connect to other elements
• All the elements are present in single level and they can be traversed in single run.	Elements can be present in multiple levels and often can't be traversed in a single run.
• Examples of linear data structures are array, stack, queue, linked list, etc	Examples of non linear data structures are tree, graph

20BCP304D

C. Space complexity and Time complexity.

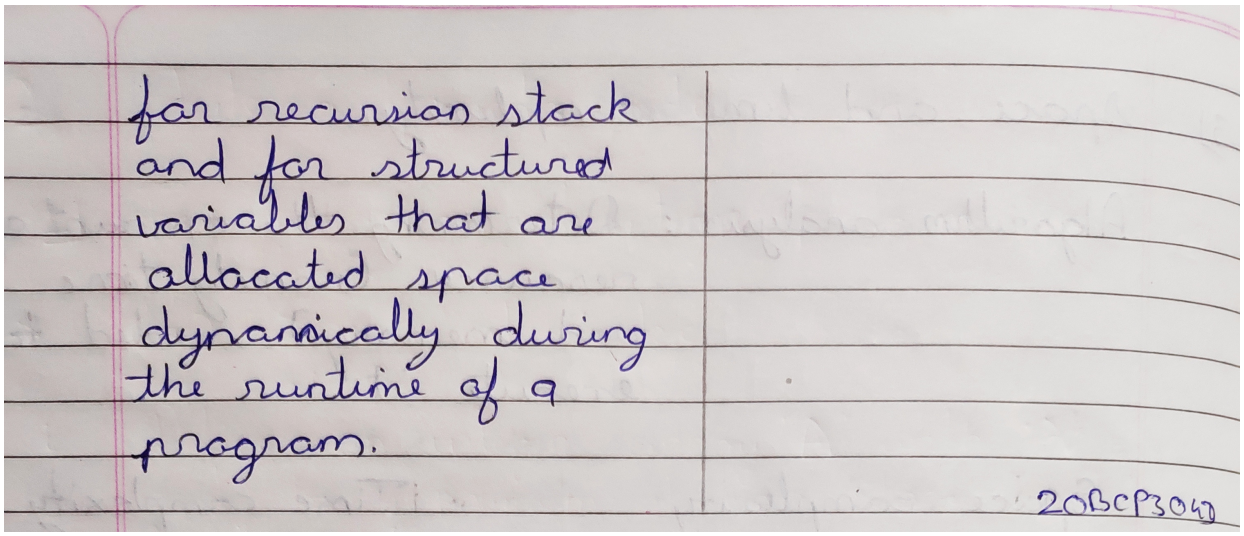


3) Space and time complexity

Algorithm analysis : Determining the amount of resources (namely time and memory) needed to execute it.

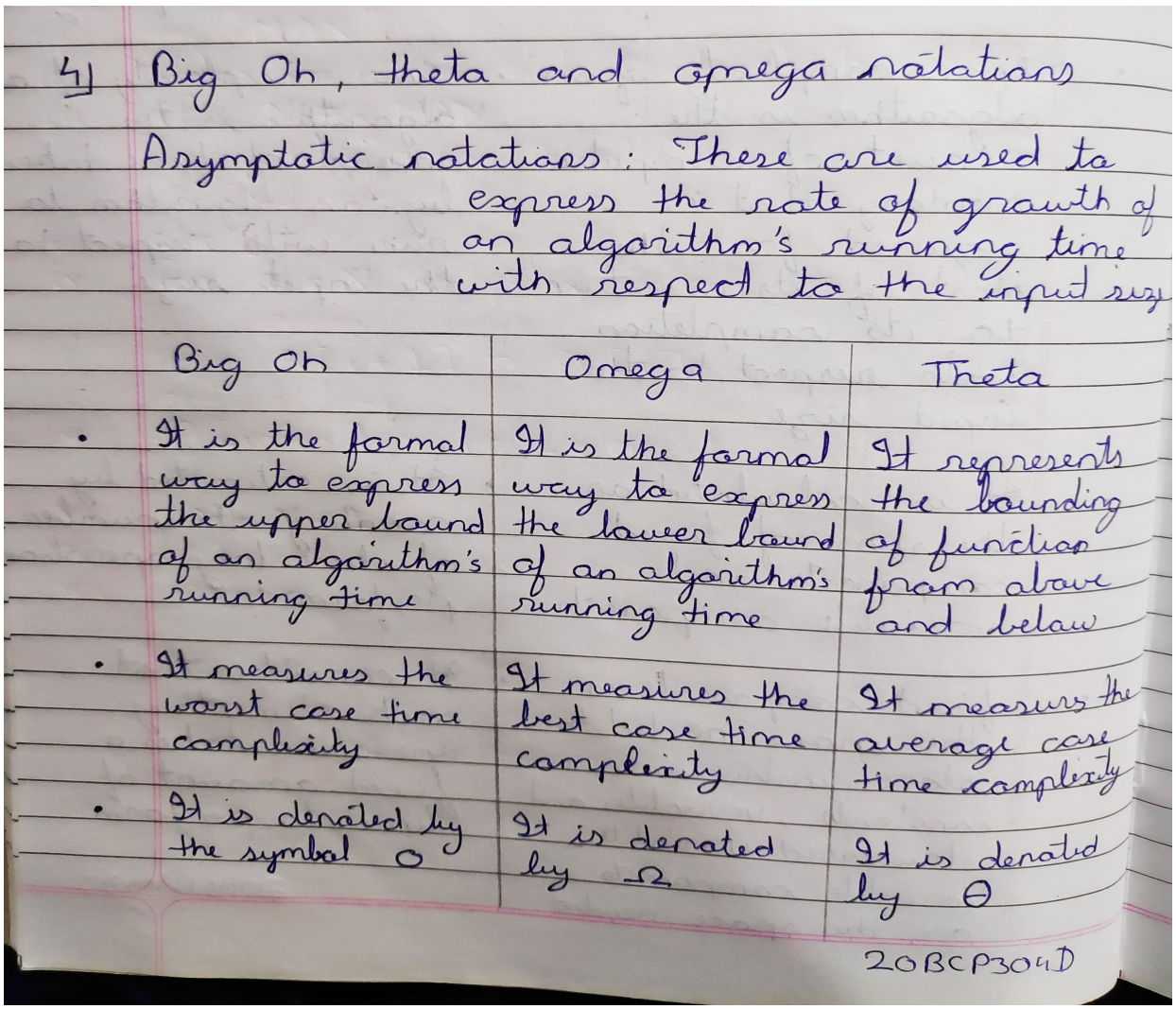
Space complexity	Time complexity
• Space complexity of an algorithm is the amount of computer memory that is required from the start of its execution to its completion with respect to the input size.	• Time complexity of a algorithm is the amount of time taken by an algorithm to run with respect to the input size
• It is calculated as the sum of fixed components and variable components	• It is estimated by counting the number of elementary operation performed
• fixed components is the space needed to store instructions, constants, variables and structured variables. Variable components are the space needed	• It considers that each elementary operation takes a fixed amount of time to perform

20BCP304D



for recursion stack and for structured variables that are allocated space dynamically during the runtime of a program.

D. Big Oh, Theta and Omega notations.

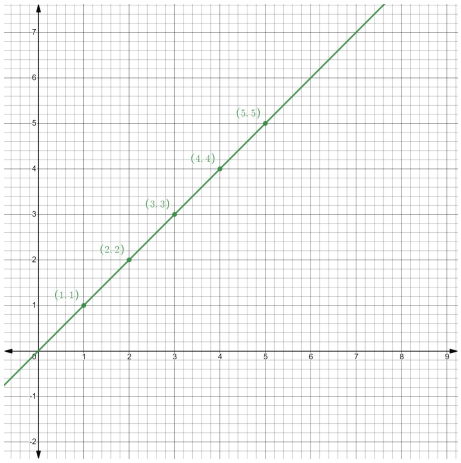
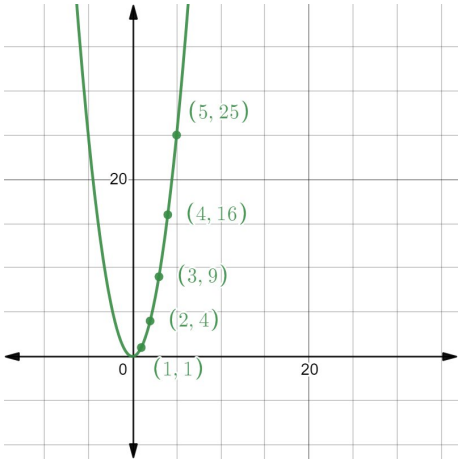
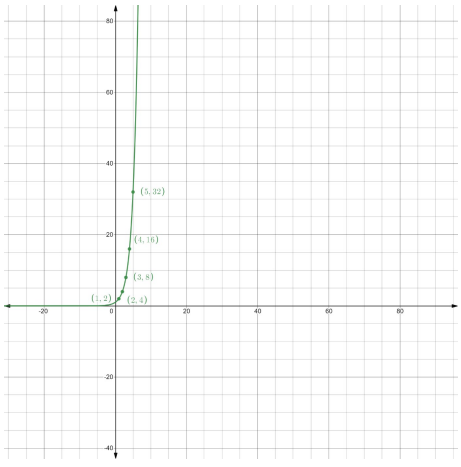


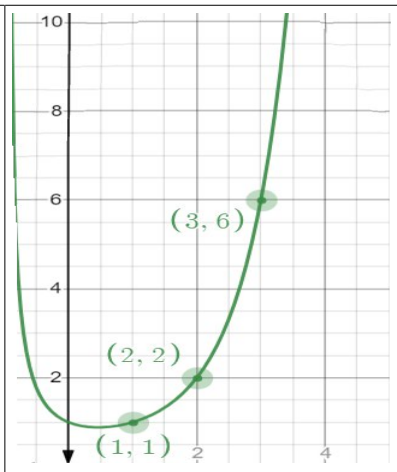
4] Big Oh, theta and omega notations

Asymptotic notations: These are used to express the rate of growth of an algorithm's running time with respect to the input size

Big Oh	Omega	Theta
• It is the formal way to express the upper bound of an algorithm's running time	It is the formal way to express the lower bound of an algorithm's running time	It represents the bounding of function from above and below
• It measures the worst case time complexity	It measures the best case time complexity	It measures the average case time complexity
• It is denoted by the symbol O	It is denoted by Ω	It is denoted by Θ

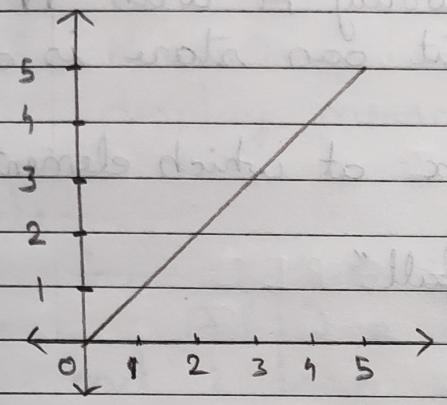
Q2. Draw graph for comparing the following function, where *n* varies from 0 to 5 (*x-axis* representing the *n* and *y-axis* representing *f(n)* function).

$f(n) = n$	
$f(n) = n^2$	
$f(n) = 2^n$	

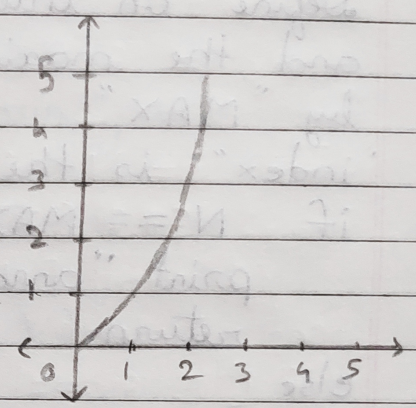
f(n) = n!	

Q2 Draw graphs for comparing the following functions, where n varies from 0 to 5 (x axis representing n and y axis representing f(n) function).

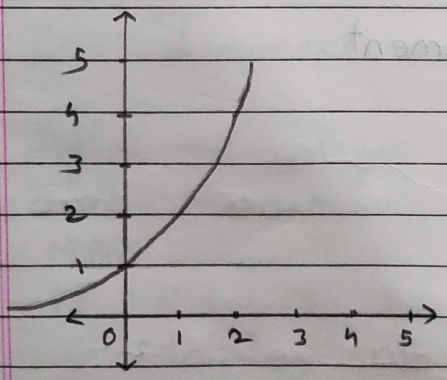
1 $f(n) = n$



2. $f(n) = n^2$



3 $f(n) = 2^n$



4. $f(n) = n!$