Tshwane University of Technology

We empower people

Student Number	
Surname & Initials	

YEAR: 2021
SEMESTER: 1
ASSESSMENT: SUMMATIVE ASSESSMENT

SUBJECT NAME:	DATA STRUCTURES AND ALGORITHMS V
SUBJECT CODE:	DTD117V
QUALIFICATION(S):	ADRS20 ADVANCED DIPLOMA IN COMPUTER SCIENCE

PAPER DESCRIPTION: COMPUTER BASED **DURATION:** 4 HOURS **PAPER:** ONLY

SPECIAL REQUIREMENTS

- ☐ **NONE**
- ☒ **NON-PROGRAMMABLE POCKET CALCULATOR**
- ☐ **SCIENTIFIC CALCULATOR**
- ☐ **COMPUTER ANSWER SHEET**
- ☐ **GRAPH PAPER**
- ☐ **DRAWING INSTRUMENTS**

OTHER: COMPUTER

INSTRUCTIONS TO CANDIDATES: ANSWER ALL QUESTIONS

THIS TEST IS TO BE ANSWERED ON PC AND ON ANSWER BOOK.
WHEN YOU ARE DONE, GIVE YOUR ANSWER SCRIPT BACK TO THE INVIGILATOR AND SUBMIT YOUR FILE.

TOTAL NUMBER OF PAGES INCLUDING COVER PAGE: 4

TOTAL NUMBER OF ANNEXURES: 0

EXAMINER: Dr N Baloyi, Prof S Mokwena & Mr N Dlamini

MODERATOR: Ms Mpho Nkosi

FULL MARKS: 100

TOTAL MARKS: 100

STUDENT TOTAL: ___

STUDENT %: ___

Preparation

Theory

a. Use the Word to answer questions.
b. Show all steps where applicable.
c. Write legibly
d. **SAVE THE ANSWER SHEET AS PDF**
e. **YOU ARE THEN REQUIRED TO UPLOAD THE PDF OUTPUT FILE INT TO SECTION A OF THE SUMMATIVE ASSESSMENT IN BRITESPACE**.

Practical

a. Use Java/C++ to create the programs required in section B
b. Write your code legibly and provide comments
c. **TO EVIDENCE YOUR OUTPUT, CAPTURE THE OUTPUT OF ALL THE VARIOUS FUNCTIONALITY REQUIREMENTS INTO A WORD DOCUMENT (CORRECTLY NUMBERED) AND SAVE IT AS PDF AT THE END.**
d. **YOU ARE THEN REQUIRED TO UPLOAD THE PDF OUTPUT FILE AND THE ASSOCIATED CODE INTO A ZIP FILE AND UPLOAD IT TO SECTION B OF THE SUMMATIVE ASSESSMENT IN BRITESPACE.**
e. If you are not able to zip the files (PDF output file for all questions and sources files), upload each individual file int section B before submitting.

Section A – Theory **[50]**

Question 1 **[20]**

1.1 Given the following adjacency matrix, draw a weighted graph depicted the adjacency matrix. (11)

	CT	PMZ	DBN	JHB	PE	EL	PTA	PLK
CT	0	10	0	8	0	0	0	0
PMZ	10	0	5	0	27	0	0	0
DBN	0	5	0	9	6	4	0	0
JHB	8	0	9	0	0	15	0	0
PE	0	27	6	0	0	0	0	20
EL	0	0	4	15	0	0	13	0
PTA	0	0	0	0	0	13	0	12
PLK	0	0	0	0	20	0	12	0

1.2 Explain how collisions arise in Hash-tables data structures. (3)

1.3 Distinguish between tree and graph data structures. (6)

Question 2 **[15]**

2.1 Define the concept of a directed graph. (3)

2.2 Define a full binary tree. (3)

2.3 What is the formula for determining the position of the left child node? (3)

2.4 What is the formula for determining the position of the left child node? (3)

2.5 What is the formula for determining the position of the parent node? (3)

Question 3 **[15]**

3.1 Insert the values 5, 9, 7, 3, 8, 12, 6, 4, and 20 into a binary search tree drawing a different tree for each stage of the insertion. (15)

Section B – Practical/Programming **[50]**

Question 4 **[30]**

Develop a simple java or C++ program that uses integer stack and list data structures. The stack should be called **NumberStack** and list **NumberList**. Perform the following operations:

4.1 Create and initialise **NumbersStack** with the following values: 55, 46, 90, 39, 20, 13, 56, 100 and 77 using the appropriate function. Each initialisation operation should display a message such as “value XX added into stack Numbers”. (7)

4.2 Display the values in the **NumberStack** data structure. (2)

4.3 Display the size of the **NumberStack** data structure. (2)

4.4 Create the **NumberList** list data structure and initialise it by moving the top **four** numbers of the **NumberStack** data structure into the **NumberList** data structure. (12)

4.5 Display the values in the **NumberList** data structure. (2)

4.6 Display the size of the **NumberList** data structure. (2)

4.7 Increment the value of the last element in the **NumberStack** data structure by 10 and display all the values of the **NumberStack** data structure. (3)

Question 5 **[20]**

Develop a simple java or C++ program to demonstrate the use of the AVL tree data structure. Perform the following operations on the AVL:

5.1 Create and initialise an AVL tree called **NumAVL** with the following values: 5, 9, 7, 3, 8, 12, 6, 4, and 20. (5)

5.2 Display the values in the **NumAVL** tree using inorder transversal. (3)

5.3 Delete the value 12 from the **NumAVL** tree. (4)

5.4 Display the values in the **NumAVL** tree. (2)

5.5 Insert the value 1 into the **NumAVL** tree. (4)

5.6 Display the values in the **NumAVL** tree. (2)

The End