79	79%
100	

THE INDEPENDENT INSTITUTE OF EDUCATION

Seat Number: 1

ASSESSMENT ANSWER BOOK

STUDENTS ARE TO COMPLETE ALL DETAILS

Book No. 1 of 1

Test ☐ Exam ☑

Standard Sitting ☑ Deferred Sitting ☐ Supplementary Sitting ☐ Dean's Exam ☐

Discontinuation Sitting ☐

Student Number: ST10478144

Qualification and Programme Name: Diploma in Software Development (DipSD)

Module / Subject Name: Programming Logic and Design

Module / Subject Code: PRLDS121/6p/w Group: Group 1

Lecturer: Lindelwa Gumede

Campus Examination Centre: Capsicum Culinary Studio (Rosebank) Venue: HSM

Due Date: 24/06/2025 Date Submitted: 24/06/2025

STUDENT DECLARATION

I declare that I have conformed to all policies and assessment rules, and have not engaged in any academically dishonest behaviour.

Signed: [signature] Date: 24/06/2025

QUESTION	01	02	03	04	05	06	07	08	09	10
MARKER										
MODERATOR										

QUESTION	11	12	13	14	15	16	17	18	19	20
MARKER										
MODERATOR										

Marker's Name: _____ Moderator's Name: _____

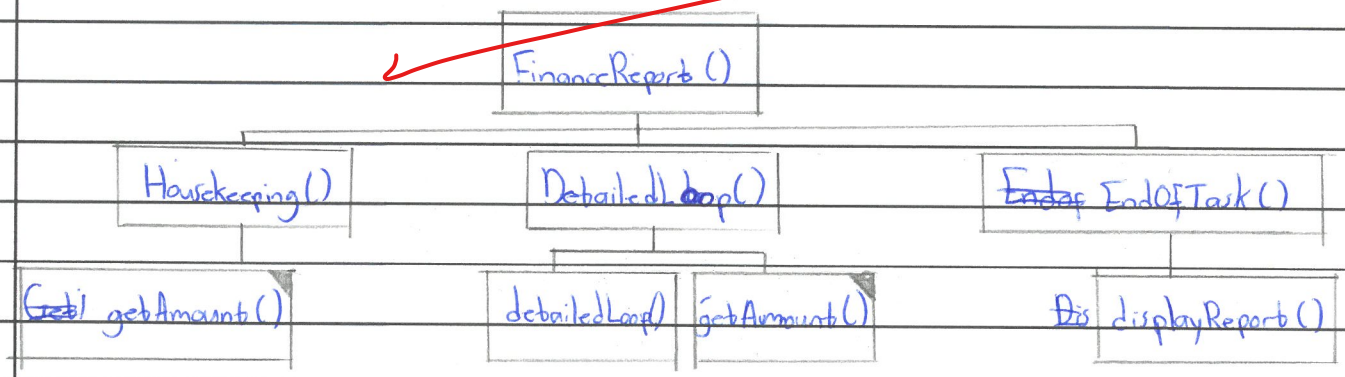
Marker's Signature: _____ Moderator's Signature: _____

Marker's Date: _____ Moderator's Date: _____

Question 1

1.1. Hierarchical charts are a chart that gives an overall picture of how a program and methods/modules work with each other. Just like in a work place there is a chart that begins at the top of the chart with "CEO", then showing the departments below it such as management, e.g. Financial department, Human Resources department, Manufacturing department, Logistics department etc. Then below each department are the workers of the individual department. Hierarchy charts are similar. It begins with the main program name, below it is the different methods or modules e.g. Housekeeping(), Detailed loop(), End of Task(). Then below each modules are the inner modules such as GetInput(), which will be under Housekeeping(), etc. All are connected. The reason to use them is because this can help the organisation or client to understand how the different modules or methods work together. They are not for showing them the complete logic of a programe.

An example of a hierachy chart.



1.2.1. Decisions ◇ symbol
1.2.2. Processing symbol
1.2.3. Input/Output symbol
1.2.4 Flowline Symbol
1.2.5. Start/Stop symbol

Question 2

2.1. Declarations

~~patientaddress~~ StringPatientAdd = "123 Bidreed Avenue" // Add is Address

2.2. The AND operator has higher ~~precedence~~, it will be ~~evaluated~~ first.

2.3. A nested loop is loop that ~~is~~ has another inside the loop. Programmers call this ~~type~~ of loop a inner loop, because it is a loop inside the main loop. ~~Cool~~ Consider the following pseudocode :-

```
START
    Declarations
    String QUIT = "1"          | num price
    num ageLimit = 20          | num discount = 0,25
    num age
    Output "Welcome to Youth Service"
    Output "Please enter your age or ", QUIT, " to exit the program"
    Input age

    WHILE age <> QUIT
        count = 0
        WHILE count < ageLimit    // This is a inner loop(A loop inside the loop)
            price = price - (age* discount)
            count = count + 1
        ENDWHILE
    ENDWHILE

    Output "Your age is", age, "Your price is", price

    STOP
```

Nested loops are loops inside a loop.

2.4.

START

Declarations

num Sum = 0

Output "Welcome to sum program"

FOR i = 1 TO 6 STEP 1

Sum = Sum + Sum

Output Sum

STOP

Question 3

3.1. START

Declarations

String QUIT = "1"

~~ITEMS[4]~~ String ITEMS[4] = "25", "34", "45", "66"

String ORDER_NUMS[4] = "or559", "or560", "or561", "or562"

num SIZE = 4

~~H~~housekeeping()

WHILE ITEM <> QUIT

detailedLoop()

ENDWHILE

endOfJob()

STOP

~~Hou~~ housekeeping()

Output "Enter the item or ", QUIT " to quit"

Input ITEM

RETURN

detailedLoop()

foundIt = 'Y' // Flag variable

~~notFound = 'N'~~ Count = 0

WHILE ~~notFound~~ ~~foundIt~~ Count <> ~~ITEMS~~ Count < SIZE - 1

Output "Our Items#:", ITEMS[Count]

Output "Their ~~ord~~ order numbers:", ORDER_NUMS[Count]

Count = Count + 1

EndWHILE

notFound = 'N'

~~If no Input~~ Output "~~Has~~ Was the ITEM youre looking for found"

~~Opt~~ Output "Please enter ", foundIt, "If found or ", notfound,"

If not found"

Input

RETURN

endOfJob()

For i=0 TO 3 STEP 1

IF Input == foundIt THEN

Output "The item you were looking for", ITEMS[i]

Output "The order number", ORDED_NUMBER[i]

ENDFOR

~~ELSE~~

ELSE

Output "Sorry, your item was not found"

RETURN

3.2. Bubble Sort is a type of sorting program that can be used to sort values from ascending order to descending order. This is how it works.

If ~~you~~ are given a list such "10,20,30,40" and ~~want to rearrange~~ the list ~~from descending~~ ascending order to descending order then the last number 40 will go up the list like a bubble (which is where the term comes from). Logically what happens is that the program will ~~first~~ compare "10" and "20" to see if the first ~~adj~~ value ~~and its adjacent value~~ is less then it's adjacent value. The answer is yes, "20" is bigger then "10" then ~~it~~ swapps the value using a swap logic or swap method or module. In that swap module or method, the program takes the ~~value~~ value assigned in ~~vath~~ the variable value1, which is 10 and assigns it to a temporary variable (called temp). Now

the variable valueOne is empty. Then the value in variable valueTwo is "20". Variable valueTwo is now assigned to variable valueOne, meaning the value "20" is now in the position value "10" was. Now because variable valueTwo was assigned to variable valueOne, variable valueTwo is empty. Temp is now assigned to variable valueTwo, meaning the value "10", which was in Temp is now assigned to valueTwo.

The two numbers "10" and "20" are now swapped. The program must be run until the desired list is accomplished. Until "40" is now first, "30" second, "20" third and "10" is fourth.

3&3. A method header must not have any spaces in between new words of the first description of the method header. The header must have a good and reasonable name. A set of parenthesis must follow after the given name. If requested, a parameter with the arguments or inside the parenthesis, which is not a must but if needed it will be done.

Question 4

4.1. Public access means everything that is in that class or method (all data fields or local variables) is accessable when it is called by it's client. The data can be altered or changed when working with it's client that has called the method/object in a class. The data is easily and accessable and is available.

Private access means all the the data fields or local variables are limited inside the class and the data cannot be altered or changed when it is called by it's client.

Private access is used with encapsulation as it makes the objects, attributes (methods) and polymorphism encapsulated or hidding hidden in the class and all available only in the class.