ICTPRG302

Apply introductory programming techniques

Assessment 1 of 6

Short Answer Questions

Assessor Guide

Version 1

Assessment Instructions

Task Overview

This assessment task is divided into ten (10) short answer questions. Read each question carefully before typing your response in the space provided.

Important: Before commencing your work, you must update your *Student name* and *Student number* in the footer from **page 2** onwards.

Additional Resources and Supporting Documents

To complete this assessment, you will need:

- Learning Material

Assessment Information

Submission

You are entitled to three (3) attempts to complete this assessment satisfactorily. Incomplete assessments will not be marked and will count as one of your three attempts.

All questions must be responded to correctly to be assessed as satisfactory for this assessment.

Answers must be typed into the space provided and submitted electronically via the Learning Platform. Hand-written assessments will not be accepted unless previously arranged with your assessor.

Reasonable adjustment

Students may request a reasonable adjustment for assessment tasks.

Reasonable adjustment usually involves varying:

- the processes for conducting the assessment (e.g. allowing additional time)
- the evidence gathering techniques (e.g. oral rather than written questioning, use of a scribe, modifications to equipment)

However, the evidence collected must allow the student to demonstrate all requirements of the unit.

Refer to the Student Handbook or contact your Trainer for further information.

Please consider the environment before printing this assessment.

Question 1

Answer the following questions about language data types, operators, expressions, and variables by filling out the four (4) tables below.

Assessor instructions: Students must answer the following questions about language data types, operators, expressions, and variables by filling out the tables below.

The acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

A. Language Data Types	
a. Identify at least three language data types used in creating codes. b. In your own words, briefly describe each identified language data type used in creating codes.	
Language Data Types Used in Creating Codes **Assessor instructions**: The student must identify at least three language data types used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to categories of data that the program can store and manipulate.	**Description of Identified Language Data Types** [Approximate word count: 30 – 50 words] **Assessor instructions**: In their own words, students must briefly describe each identified language data type used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to a description of the characteristics of each data type to help the developers understand their purpose and applicability.
i. Responses can include: • Integer • Float or double • String • Character	For example, if the student identified Integer as a language data type used in creating codes, their response can be: An integer denotes complete numbers, does not include fractions, and encompasses both positive and negative values. It also proves valuable for counting and representing distinct, discrete quantities in various applications.
ii.	
iii.	

B. Operators

a. Identify at least three types of language operators used in creating codes.

b. In your own words, briefly describe each identified language operator used in creating codes.

Types of Operators Used in Creating Codes	Description of Identified Operators [Approximate word count: 30 – 50 words]
Assessor instructions: The student must identify at least three types of operators used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to symbols or keywords that perform operations to produce results.	**Assessor instructions**: In their own words, students must briefly describe each identified type of operator used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to a description of the characteristics of each type of operator used in different aspects of programming tasks.
i. Responses can include: ▪ Arithmetic ▪ Assignment ▪ Bitwise ▪ Comparison	For example, the student identified Arithmetic as a type of operator used in creating codes. The student's response can be: Arithmetic operators are fundamental tools in coding that perform mathematical operations on numeric values. They include addition (+), subtraction (-), multiplication (*), division (/), and modulus (%). These operators follow standard mathematical rules and help in calculations, data manipulation, and numeric analysis within programs.
ii.	
iii.	

C. Expression

a. Identify at least three types of language expression used in creating code.

b. In your own words, briefly describe each identified language expression used in creating codes.

Types of Expression Used in Creating Codes	Description of Identified Expression [Approximate word count: 30 – 50 words]
Assessor instructions: The student must identify at least three types of expression used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to expression elements that represent computations or calculations within a program.to manipulate and transform data.	**Assessor instructions**: In their own words, students must briefly describe each identified type of expression used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to a description of the characteristics of each type of expression used in writing functional code in various programming languages.
i. Responses can include: ▪ String concatenation ▪ Comparison expression ▪ Logical expression ▪ Arithmetic	For example, the student identified String concatenation as a type of expression used in creating codes. Student response can be: String concatenation is a coding technique that involves combining multiple strings to create a single, longer string. It is achieved using the concatenation operator (+) or specific string manipulation methods provided by the programming language. This technique is essential for creating dynamic text output, constructing messages, and formatting data with variable values.
ii.	
iii.	

D. Variables

a. Identify at least three types of language variables used in creating codes.

b. In your own words, briefly describe each identified language variable used in creating codes.

Types of Variables Used in Creating Codes	Description of Identified Variables [Approximate word count: 30 – 50 words]
Assessor instructions: The student must identify at least three types of variables used in creating codes. Although responses will vary, for satisfactory performance, the student's response must correspond to categories of variables commonly used in programming to store and manipulate data during program execution.	**Assessor instructions**: In their own words, students must briefly describe each identified type of variable used in creating codes. The student's response must be in 30 words or more. Although responses will vary, for satisfactory performance, the student's response must correspond to a description of the characteristics of each type of variable for structured programming.
i. Responses can include: ▪ Constant variables ▪ Global variables ▪ Class variables ▪ Instance variables	For example, the student identified Constant variable as a type of the variable used in creating codes. The student's response can be: These are variables whose values cannot be changed once they are assigned. They are usually declared using the 'const' keyword. Likewise, these represent fixed values that remain constant throughout the program.
ii.	
iii.	

Question 2

Complete the table below about basic language syntax rules.

- Identify at least three basic syntaxes of a programming language.
- Identify at least three basic syntax rules applied to **each of the identified basic syntaxes** of the programming language.
- In your own words, briefly describe each basic syntax rule identified corresponding to its basic syntax.

[Approximate word count: 30 – 50 words]

Assessor instructions: Students must complete the tables.

The acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

Basic Syntax of Programming	Basic Syntax Rules of Identified Basic Syntax	Description of the Identified Basic Syntax Rules
Assessor instructions: The student must identify at least three basic syntaxes of a programming language. Although responses will vary, for satisfactory performance, the student's response must correspond to concepts that define the format of the expression in a programming language.	**Assessor instructions**: The student must identify at least three basic syntax rules applied to **each** of the identified basic syntaxes of the programming language. Although responses will vary, for satisfactory performance, the student's response must correspond to conventions to follow when referring to a specific element defined by the language syntax.	[Approximate word count: 30 – 50 words] **Assessor instructions**: In their own words, students must briefly describe each basic syntax rule identified corresponding to its basic syntax. Although responses will vary, for satisfactory performance, the student's response must correspond to the characteristics of each identified basic syntax rule to provide insights into programming concepts.
a. Identifiers	i. Responses can include: - Valid Characters - Length Limitations - Naming Conventions - Reserve Identifiers	For example, the student identified Valid Characters as a basic syntax rule for identifiers. Student response can be: Identifiers typically consist of letters (uppercase and lowercase), digits and underscores (_). However, some languages may have extra rules, such as allowing certain special characters. Likewise, they can disallow digits as the first character.

	ii.	
	iii.	
Basic Syntax of Programming	**Basic Syntax Rules of Identified Basic Syntax**	**Description of the Identified Basic Syntax Rules**
b. Keywords	i. Responses can include: ▪ Case Sensitivity ▪ Reserve Usage ▪ Spelling Specific	For example, the student identified Case Sensitivity as a basic syntax rule for keywords. Student response can be: Keywords are typically case-sensitive. It means that they must be written exactly as specified by the programming language. For example, 'if' and 'IF' may be treated as two different keywords.
	ii.	
	iii.	
Basic Syntax of Programming	**Basic Syntax Rules of Identified Basic Syntax**	**Description of the Identified Basic Syntax Rules**
c. Literals	i. Responses can include: ▪ Numeric ▪ String ▪ Boolean ▪ Character	For example, the student identified Numeric as a basic syntax rule for literals. Student response can be: These numerical values can be integrated into various mathematical equations, formulas, or operations. By doing so, you can perform

		calculations, comparisons, and other mathematical manipulations using these values, which is a fundamental aspect of programming for various applications.
	ii.	
	iii.	

Question 3

Answer the following questions about the sequence, selection, and iteration constructs when applying language syntax.

Assessor instructions: Students must answer the following questions about the sequence, selection, and iteration constructs when applying language syntax..

The acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

A. Sequence Constructs
a. In your own words, explain the importance of sequence constructs when applying language syntax. [Approximate word count: 30 – 50 words] **Assessor instructions**: In their own words, the students must explain the importance of sequence constructs when applying language syntax. Although responses will vary, for satisfactory performance, the student's response must correspond to the role of sequence constructs in the logical order of operations and the ability to control the execution flow in programming.
Sequence constructs define the logical flow of operations in a program. Organising statements in a specific sequence can control the execution order. They ensure that each statement is processed in the desired sequence.
b. In your own words, briefly explain the given common practices to apply language syntax in sequence constructs. [Approximate word count: 30 – 50 words].

In their own words, the students must briefly explain the given common practices to apply language syntax in sequence constructs.

Although wording will vary, for satisfactory performance, the student's response must correspond to an explanation of guidelines to follow for each practice to achieve the desired logic and functionality in programming.

Common Practices Used in Sequence Constructs	Explanation of the Common Practices Used in Sequence Constructs
i. Follow the correct order of statements.	Sequential code executes statements in the order they appear. You must arrange your statements in the desired order to achieve the desired sequence of actions. Be mindful of any dependencies between statements and ensure they are executed in the correct order.
ii. Use appropriate syntax for data types and variables	When working with variables and data types, use the syntax specified by the language. Declare variables using the correct syntax and follow the naming conventions for data types and variables.
iii. Check for syntax errors	You must follow the syntax rules strictly to avoid syntax errors. Syntax errors occur when code does not conform to the expected syntax of the language. Check for error messages or warnings generated by your code editor or compiler. Moreover, you should fix any syntax errors before executing your program.

B. Selection Constructs

a. In your own words, explain the importance of selection constructs when applying language syntax.

[Approximate word count: 30 – 50 words]

Assessor instructions: In their own words, the students must explain the importance of selection constructs when applying language syntax.

Although responses will vary, for satisfactory performance, the student's response must correspond to the role of selection constructs in governing the flow of code based on conditions and program execution.

Selection constructs are basic language features allowing decision-making and branching execution paths. They are based on certain conditions. Likewise, they are crucial in controlling the flow of a program and determining which actions to perform.

b. In your own words, briefly explain the given common practices to apply language syntax in selection constructs.

[Approximate word count: 25 – 50 words].

Assessor instructions: In their own words, the students must briefly explain the given common practices to apply language syntax in selection constructs.

Although wording will vary, for satisfactory performance, the student's response must correspond to an explanation of guidelines to follow that aim to improve the quality and readability of code within selection constructs.

Common Practices Used in Selection Constructs	Explanation of the Common Practices Used in Selection Constructs
i. Use descriptive and meaningful conditionals.	Ensure the conditions in your selection constructs are clear and easy to understand. Use variable names and logical operators that accurately represent the intended logic.
ii. Keep conditions simple and concise.	Avoid unnecessary complex conditions. Break down complex conditions into smaller, more manageable parts using Boolean logic operators like 'AND [&&]' and 'OR [||]'. It helps improve code readability and makes it easier to test and debug
iii. Format your code properly	Use proper indentation and formatting to make your code more readable. Indent the code within the selection construct's code block to see it from the surrounding code. Stable indentation improves code readability and helps identify the code's structure.

C. Iteration Constructs

a. In your own words, explain the importance of iteration constructs when applying language syntax.

[Approximate word count: 30 – 50 words]

Assessor instructions: In their own words, the students must explain the importance of iteration constructs when applying language syntax.

Although responses will vary, for satisfactory performance, the student's response must correspond to role of iteration constructs to enhance efficiency and reduce redundancy.

The iteration constructs automate repetitive tasks and perform operations iteratively. They provide a way to handle scenarios where you must process a block of code many times without duplicating the code manually. It can streamline and optimise your code.

b. In your own words, briefly explain the given common practices to apply language syntax in iteration constructs.

[Approximate word count: 40 – 60 words]

Assessor instructions: In their own words, the students must briefly explain the given common practices to apply language syntax in iteration constructs.

Although wording will vary, for satisfactory performance, the student's response must correspond to an explanation of guidelines to follow that aim to improve the quality and readability of code within selection constructs.

Common Practices Used in Iteration Constructs	Explanation of the Common Practices Used in Iteration Constructs
i. Initialise loop variables before entering the loop	When initialising loop variables, it's crucial to use meaningful names and correct data types, set initial values that match the loop's purpose, clear prior values to prevent errors, align with loop logic, account for termination conditions, and document the process. This practice ensures a dependable starting point, boosts code clarity, and minimises the chance of errors or unexpected outcomes.
ii. Create loop conditions that accurately represent the desired termination condition	To ensure loops end as intended and avoid problems like infinite loops, it's important to craft clear and logical termination conditions. This entails creating expressions aligned with defined criteria, including relevant variables, rigorously testing for correctness and edge cases, and documenting the conditions comprehensively.
iii. Keep the code within the loop body concise and focused	Highlighting the significance of concise and focused code within the loop's block is crucial. By trimming needless elements and staying focused on the loop's intended tasks, you enhance code clarity, streamline debugging, and foster efficient execution, resulting in more effective and manageable programming.

Question 4

In your own words, outline the ten development process for small-sized applications in chronological order..

Assessor instructions: Students must outline the ten development processes for small-sized applications in chronological order.

Although wording will vary, for satisfactory performance, the student's response must correspond to the specific processes in developing small-sized applications that occur in chronological order.

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

a. Understand the objectives and requirements of the application. Identify the target audience and key functionalities needed.
b. Plan the architecture and user interface of the application. Create wireframes or mock-ups to visualise the layout and flow.
c. Write the code for the application based on the design and requirements. Develop each application task or functionality
d. Thoroughly test the application to identify and fix any bugs or issues. Perform unit testing, integration testing and functional testing.
e. Prepare the application for deployment to the production environment. Set up servers and databases and ensure the application is ready for public use.
f. Have a small group of users test the application to validate its functionality and usability.
g. Make the application live and available to the target users.
h. Monitor the application's performance and gather user feedback. Make any necessary improvements or updates based on user feedback.
i. Maintain and support the application. Address any issues that arise and update the application as needed.
j. Create clear and comprehensive documents that detail various aspects of the application. These include the requirements, design, implementation and testing. It ensures that members understand the application's function, design and usage.

Question 5

Answer the following questions about industry programming standards and guidelines.

Assessor instructions: Students must answer the following questions about industry programming standards and guidelines..

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

<table>
<tr><th colspan="2">A. Industry Programming Standards</th></tr>
<tr><td colspan="2">Complete the table below about industry programming standards.
a. Provide at least three programming standards followed by the ICT industry.
b. In your own words, explain the importance of each identified standard followed by the ICT industry when writing codes.</td></tr>
<tr><td><b>Programming Standards</b>
<b>Assessor instructions</b>: Students must identify at least three programming standards the ICT industry follows.
For satisfactory performance, the student's response must correspond to the rules and conventions followed by the ICT industry when writing code.</td><td><b>Explanation of the Importance of the Identified Programming Standards</b>
[Approximate word count: 30 – 60 words]
<b>Assessor instructions</b>: In their own words, the students must explain the importance of each identified standard followed by the ICT industry to write codes.
Although responses will vary, for satisfactory performance, the student's response must correspond to the outcomes of the identified programming standard essential to enhancing code quality, readability, and maintainability.</td></tr>
<tr><td>i.
Responses can include:
▪ Naming Conventions
▪ Code formatting
▪ Documentation
▪ Comments</td><td>For example, the student identified Code formatting as a programming standard followed by the ICT industry to write codes.
Student response can be:
Code formatting is crucial in software development as it enhances readability, collaboration, and maintenance. Consistent formatting improves code comprehension, reduces errors, aids debugging, and facilitates smooth teamwork, making the development process more efficient and effective.</td></tr>
<tr><td>ii.</td><td></td></tr>
<tr><td>iii.</td><td></td></tr>
<tr><th colspan="2">B. Industry Programming Guidelines</th></tr>
</table>

Complete the table below about industry programming guidelines. a. Provide at least three programming guidelines followed by the ICT industry. b. In your own words, explain the importance of each identified standard followed by the ICT industry when writing codes.	
Programming Guidelines **Assessor instructions:** Students must identify at least three programming guidelines followed by the ICT industry. For satisfactory performance, the student's response must correspond to recommended practices, principles and suggestions followed within the ICT industry.	**Explanation of the Importance of the Identified Programming Guidelines** [Approximate word count: 30 – 60 words] **Assessor instructions**: In their own words, the students must explain the importance of each identified guideline followed by the ICT industry to write codes. Although responses will vary, for satisfactory performance, student's responses must correspond to the outcomes of the identified programming guidelines to create a clear and well-structured code for an efficient software development process.
i. Responses can include: ▪ Consistent Variable Naming ▪ Function Length ▪ Use of comments ▪ Code Documentation	For example, the student identified Consistent variable naming as a programming guideline followed by the ICT industry to write codes. Student response can be: It ensures uniformity and clarity across codebases, simplifying collaboration among developers. This practice enhances code readability, reducing confusion and errors. Moreover, consistent variable names facilitate code maintenance and scalability, fostering an organised and sustainable development process.
ii.	
iii.	

Question 6

Answer the following questions about commenting techniques based on each scenario below.

Assessor instructions: Students must answer the following questions about commenting techniques based on each scenario below.

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

Scenario 1

Jake is a developer and is working on a team developing a machine learning application that predicts customer preferences for an online shopping platform. Jake is examining a code snippet written by one of his team members, Lisa. Lisa wrote a code that calculates the total price of the items in a shopping cart and added some comments. While reviewing the code, Jake come across the following line:

```
subtotal = calculate_subtotal(cart_items)  # Calculate subtotal based on cart items
```

Identify the specific commenting technique Lisa followed when creating the comment in the scenario.

Assessor instructions: Students must identify the specific commenting technique Lisa followed when creating the comment in the scenario.

For satisfactory performance, student’s response must correspond to the benchmark answer below.

Single-line comments

Scenario 2

Mark is a full-stack developer working on an e-commerce project. Together with his team, they are developing an API for an online bookstore. Jay, his teammate, wrote a code for handling user authentication and added some comments. As Mark inspects the code he come across the following section.

```
class UserAuthentication:
  def __init__(self, username, password):
    """
    Initializes a new instance of the UserAuthentication class.

    :param str username: The username provided by the user.
    :param str password: The password provided by the user.
    """
    self.username = username
    self.password = password

  def authenticate(self):
    """
    Authenticates the user based on the provided username and password.

    :return: True if authentication is successful, False otherwise.
    :rtype: bool
    """
    # Authentication logic goes here
```

Identify the specific commenting technique Jay followed when creating the comment in the scenario.

Assessor instructions: Students must identify the specific commenting technique Jay followed when creating the comment in the scenario.

For satisfactory performance, student’s response must correspond to the benchmark answer below.

Documentation Comment

Scenario 3

Emma is a software engineer and is managing a team working on developing a financial management application. Sarah her teammate, wrote and shared a code snippet which is a function responsible for processing user input and generating a personalized welcome message. As Emma examine the code, you come across the following section:

```
def generate_welcome_message(name):
  """
  Generates a personalized welcome message for the user.


  :param str name: The name of the user.
  :return: The personalized welcome message.
  :rtype: str
  """
  welcome_message = f"Hello, {name}! Welcome to our website."
  return welcome_message


# TODO: Implement user preferences retrieval to customize welcome message further
```

Identify the specific commenting technique Sarah followed when creating the comment in the scenario.

Assessor instructions: Students must identify the specific commenting technique Sarah followed when creating the comment in the scenario.

For satisfactory performance, student’s response must correspond to the benchmark answer below.

TODO Comments

Question 7

In your own words, briefly describe at least three reasons why commenting techniques are important in clarifying the meaning of the code.

[Approximate word count: 25 – 30 words per reason]

Assessor instructions: Students must describe at least three reasons why commenting techniques are important in clarifying the meaning of the code.

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.
- Briefly describe at least three reasons why commenting techniques are important in clarifying the meaning of the code.
- Their response must correspond to the role of comments in programming as a means of communication, documentation, and guidance.

Benchmark answers are provided below.

a. Responses can include: ▪ They help collaboration among team members. Comments provide insights into the code's functionality and design choices. Likewise, they help the team communicate and share knowledge effectively. ▪ They provide information about the purpose of code segments, algorithms, or data structures. Likewise, they explain the code's behaviour and usage. This makes it easier to use and extend the codebase. ▪ They help developers understand the code logic, intentions, and implementation details. They serve as a form of self-documentation, making it easier to maintain, modify and debug code. ▪ They improve code readability by breaking down complex logic into smaller parts. Likewise, they provide context and help readers follow the code's flow and intentions.
b.
c.

Question 8

Complete the table below about debugging techniques.

a. Provide at least three debugging techniques used to detect and correct errors.

b. In your own words, describe each identified debugging techniques used to detect and correct errors.

Assessor instructions: Students must complete the table below about the current and emerging system and product trends and directions.

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

Debugging Techniques **Assessor instructions:** Students must provide at least three debugging techniques used to detect and correct errors. Although responses will vary, for satisfactory performance, student's responses must correspond to strategies that can be used for step-by-step execution of code and easier identification of logic errors.	Description of the Identified Debugging Techniques [Approximate word count: 50 - 70 words] **Assessor instructions**: In their own words, the students must describe each identified debugging technique used to detect and correct errors. Although responses will vary, for satisfactory performance, the student's response must correspond to a description of characteristics of the identified debugging technique to enhance the quality, reliability, and functionality of software code during development.
i. Responses can include: ▪ Debugging Tools ▪ Runtime error handling ▪ Interactive Debugging ▪ Conditional Debugging	For example, the student identified Debugging Tools as a debugging technique used to detect and correct errors. Student response can be: Debugging tools encompass a range of software resources used to identify and rectify errors, issues, or defects within a program's code. These tools offer features like breakpoints, step-by-step execution, variable inspection, and error tracking, aiding developers in pinpointing and resolving problems efficiently. Debugging tools are essential for diagnosing issues during the development process, enhancing code quality, and ensuring that software functions as intended.
ii.	
iii.	

Question 9

Complete the table below about application testing methods.

a. Provide at least three types of application testing methods used throughout the development process.

b. In your own words, describe each identified type of application testing method used throughout the development process.

Assessor instructions: Students must complete the table below about the current and emerging system and product trends and directions.

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

Types of Application Testing Methods **Assessor instructions**: Students must provide at least three types of application testing methods used throughout the development process. Although responses will vary, for satisfactory performance, students' responses must correspond to approaches to ensure that the app meets the requirements during the development phase.	Description of the Identified Types of Application Testing Methods [Approximate word count: 30 – 60 words] **Assessor instructions**: In their own words, the students must describe each identified type of application testing method throughout the development process. Although responses will vary, for satisfactory performance, the student's response must correspond to the description of characteristics of each identified type of application testing method to ensure that the software functions correctly.
i. Responses can include: ▪ Unit Testing ▪ Integration Testing ▪ System Testing ▪ Regression Testing	For example, the student identified Unit Testing as a type of application testing method throughout the development process. Student response can be: Unit testing is a fundamental testing technique in software development where individual units or components of a program are tested in isolation. These units can be functions, methods, or modules. Unit tests validate the correctness of these units by comparing their actual outputs to expected results.
ii.	
iii.	

Question 10

Complete the table below about basic data structures.

- Identify at least three basic data structures used to represent and manage data in a structured manner.
- In your own words, describe each basic data structure used to represent and manage data.
- Provide at least three tasks commonly performed corresponding to each identified basic data structure.

Assessor instructions: Students must complete the table below about the current and emerging system and product trends and directions.

Students are likely to use wording different from the sample answer provided. However, the acceptable responses must:

- Be within the specified word limit.
- Reflect the characteristics described in the exemplar answer.

Benchmark answers are provided below.

Basic Data Structures **Assessor instructions:** Students must identify at least three basic data structures used to represent and manage data in a structured manner. Although responses will vary, for satisfactory performance, candidate's response must correspond to frameworks used to optimise data storage and processing.	Explanation of the Identified Basic Data Structures [Approximate word count: 30 – 60 words] **Assessor instructions**: In their own words, the students must describe each identified basic data structure used to represent and manage data. Although responses will vary, for satisfactory performance, candidate's response must correspond to the descriptions of the framework used to have a more efficient solutions to coding problems.	Tasks Commonly Performed Based on the Basic Data Structure **Assessor instructions**: Students must provide at least at least three tasks commonly performed corresponding to each identified basic data structure. Although wording will vary, for satisfactory performance, candidate's response must correspond to the specific actions performed using each identified basic data structures to aid in solving problems.
a.		i.
		ii.
		iii.
b.		i.
		ii.
		iii.

c.		i.
		ii.
		iii.
a. Arrays	An array is a set of elements of the same data type stored at contiguous memory locations. They provide a way to store and access many elements using a single variable. They have a fixed size and allow random access to elements using an index.	Responses must include at least three of the following: ▪ Storing lists of items ▪ Implementing matrices ▪ Handling Large amounts of data ▪ Representing ordered collections
b. Lists	Lists are dynamic data structures that can store elements of different data types and can grow or shrink in size. On the other hand, an array list is a specific implementation of lists in some programming languages like Java. They provide resizable arrays and offer operations for adding, removing, and accessing elements.	Responses must include at least three of the following: ▪ Managing dynamic collection of data ▪ Implementing data structures like stacks and queues ▪ Implementing algorithms like sorting and searching ▪ Representing data in a structured format aiding in data display.t
c. Sets	A set is a group of unique elements with no order. They enforce uniqueness. It means they cannot contain duplicate elements. Likewise, they provide operations for set operations like union, intersection, and difference.	Responses must include at least three of the following: ▪ Removing duplicates from a collection ▪ Checking for the presence of an element ▪ Implementing mathematical set operations ▪ Eliminating ordering in scenarios where order doesn't matter.

d. Trees	A tree is a hierarchical data structure composed of nodes that have parent-child relationships. They represent hierarchical relationships between elements. Trees can be binary or have many child nodes. Binary nodes can have at most two child nodes. This means that each node in a binary tree can have either zero, one or two child nodes.	Responses must include at least three of the following: ▪ Organising hierarchical data ▪ Implementing efficient search and retrieval algorithms ▪ Representing file systems and directories ▪ Representing data for various graph algorithms.

Assessment submission checklist

Students must have completed all questions within this assessment before submitting. This includes:

1	Ten (10) short answer questions completed in the spaces provided.	□

Assessment feedback

Assessors are to indicate the assessment outcome as Satisfactory [S] or Not Yet Satisfactory [NYS].

Assessor comments:	□ S	□ NYS

Congratulations, you have reached the end of Assessment 1!