|  |
| --- |
| **FURTHER EDUCATION AND TRAINING CERTIFICATE: INFORMATION TECHNOLOGY: SYSTEMS DEVELOPMENT**  **ID 78965 LEVEL 4 – CREDITS 165** |
| **LEARNER WORKBOOK**  **SAQA: 14918**  **DESCRIBE THE PRINCIPLES OF COMPUTER PROGRAMMING** |

**Learner Information:**

|  |  |
| --- | --- |
| **Details** | **Please Complete this Section** |
| Name & Surname: | Mila Mihlali Ngewu |
| Organisation: | Nelson Mandela Bay iHUB |
| Unit/Dept: | ICT |
| Facilitator Name: | Anneline Nombewu |
| Date Started: | 18/09/2023 |
| Date of Completion: | 19/09/2023 |

**Copyright**

All rights reserved. The copyright of this document, its previous editions and any annexures thereto, is protected and expressly reserved. No part of this document may be reproduced, stored in a retrievable system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior permission.

**OVERVIEW**

|  |  |  |  |
| --- | --- | --- | --- |
| **About the Learner Workbook** | This Learner Exercise Workbook has been designed and developed to evaluate learners’ level of understanding of the  **Describe the principles of Computer Programming.** It forms part of a series of Learner Workbooks that have been developed for **FURTHER EDUCATION AND TRAINING CERTIFICATE: INFORMATION TECHNOLOGY: SYSTEMS DEVELOPMENT ID 78965 LEVEL 4 – CREDITS 165** | | |
| **Purpose** | The purpose of this Learner Exercise Workbook is to evaluate learners understanding on the specific outcomes and/or assessment criteria of the following SAQA Registered Unit Standards: | | |
| **US No** | **US Title** | **Level** | **Credits** |
| **14918** | **Describe the principles of Computer Programming** | **3** | **5** |
| **Context** | This assessment represents the Formative Assessment component of the **FURTHER EDUCATION AND TRAINING CERTIFICATE: INFORMATION TECHNOLOGY: SYSTEMS DEVELOPMENT**  **ID 78965 LEVEL 4 – CREDITS 165** and should be completed in the classroom/training room. | | |
| **Resources** | The following are resources needed for this assessment:   1. Learner Guide; and 2. Assessment Preparation. | | |
| **Instructions to Facilitators** | Facilitators will be required to:   * Explain the completion of the workbook to each learner; and * Interview the learner on similar questions, should he/she not be able to write. | | |
| **Instructions to Learners** | Learners will be required to:   * Complete the workbook as per the instructions; * Ensure that all questions are completed; * Ensure that the completion of the workbook is their own work; * Ensure that all annexure are attached to the workbook and clearly referred to; | | |
| **Assessment Time** | Learners are required to complete this assessment within the allocated time frame of.... hours. | | |
| **Total Mark** | This formative assessment carries a total mark of **\_ points**. In order to meet the pass mark, learners are required to achieve a minimum of **80%** of the total marks. | | |
| **Equipment** | Learners are required to have the following equipment in order to complete this workbook:   * Pen and Pencil; * Ruler; and * Exam Pad – for additional paper. | | |

# **GENERAL INFORMATION**

|  |  |  |
| --- | --- | --- |
| **LEARNER DETAILS** | | |
| **Learner Full Names** | **Mila Mihlali Ngewu** | |
| **Learner ID No.:** | **9909106615084** | |
| **Organisation:** | **Nelson Mandela iHUB** | |
| **Unit/Dept:** | **ICT** | |
| **Contact Details:** | **Telephone /Cell Numbers:** | **Email Address:** |
| **082 365 5804** | **ngewumila007@gmail.com** |
| **WORKSHOP DETAILS** | | |
| **Workshop Venue:** | **NMB iHUB** | |
| **Facilitator Name:** | **Anneline Nombewu** | |
| **Date Started:** | **18 September 2023** | |
| **Date Completed:** | **19 September 2023** | |

# **ASSESSMENT PREPARATION CHECKLIST**

|  |  |  |  |
| --- | --- | --- | --- |
| **DESCRIPTION** | **YES** | **NO** | **COMMENTS/CONTINGENCY** |
| This assessment is a formative assessment and it is based on the outlined unit standard/s for the **Describe the principles of Computer Programming** module. |  |  |  |
| Your assessment evidence for **Describe the principles of Computer Programming** module needs to be submitted on....... (day) of...............(month)...........(year) at the following address/place................................................................ |  |  |  |
| You will be assessed based on the outlined Unit Standards. The assessment activities are linked to specific outcomes/assessment criteria of the outlined Unit Standards. |  |  |  |
| To determine your competence level, the following are the methods to be used for this assessment:   1. ..................................................... 2. ..................................................... |  |  |  |
| * To be declared competent on **Describe the principles of Computer Programming** module (formative assessment), you should have obtained at least 80% of the total mark of this assessment. |  |  |  |
| You will be provided with detailed feedback on your performance of this assessment as follows:   1. Written Feedback 2. Verbal Feedback |  |  |  |
| Should you be declared “not yet competent” on this assessment, you will be entitled for re-assessment opportunity/ies. |  |  |  |
| You will be required to re-submit evidence (only for areas) you were declared not yet competent. A date for re-submission will be agreed with the assessor. |  |  |  |
| You will be entitled to lodge an appeal should you not be satisfied with the assessment decision of your assessment. |  |  |  |
| You will be required to provide the assessor feedback on assessment procedure – this is to assist in improving the assessment practices. |  |  |  |
| Your results of assessment and portfolio of evidence information will not be provided to any person without your written consent. |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Learner’s Declaration** | | | |
| I Mila Ngewu herewith declare that I am ready for the assessment, that we have reviewed the assessment preparation and plan, I understand the assessment process and I am happy that the assessment will be conducted in a fair manner. | | | |
| **Learner Signature:** | **Date:** | **Facilitator Signature:** | **Date:** |
|  | **18/09/2023** |  | **18/09/2023** |

Learning Unit1

**UNIT STANDARD NUMBER :** 14918

**Describe the principles of Computer Programming**

**LEVEL ON THE NQF :** 3

**CREDITS :** 5

**FIELD :** Physical, Mathematical, Computer and Life Sciences

**SUB FIELD :** Construction Information Technology and Computer Sciences

|  |  |
| --- | --- |
| **PURPOSE:** | This unit standard is intended:  • to provide a conceptual knowledge of the areas covered  • for those entering the workplace in the area of systems development  • as additional knowledge for those wanting to understand the areas covered  People credited with this unit standard are able to:  • describe problem analysis and program design techniques  • describe different data representations used in computer programs  • describe basic programming principles  • described the principles used in designing a computer program  The performance of all elements is to a standard that allows for further learning in this area |
| **LEARNING ASSUMED TO BE IN PLACE:** | |
| Open.  The credit value of this unit is based on a person having the prior knowledge and skills to:  • Demonstrate an understanding of fundamental mathematics (at least NQF level 3)  • Demonstrate PC competency skills (End User Computing unit standards up to Level 3). | |

|  |
| --- |
| **SESSION 1.**  **Describe problem analysis and program design techniques.** |
| **Learning Outcomes** |
| * The description provides an appreciation of the steps and techniques of program maintenance. * The description identifies different problem analysis techniques (at least 2). * The description identifies different programming design techniques. |

|  |  |
| --- | --- |
| http://3.bp.blogspot.com/_0EodaYtqevU/TMun5XOj03I/AAAAAAAAAIU/lzrnWelQjgc/s1600/group-discussion.jpg | **Answer the following questions according to the instructions provided** |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **1** | **List the steps and techniques of program maintenance.** | **5** |

1. Understand the problem.

2. Plan the logic of the program.

3. Code the program using a structured high level computer language.

4. Using a compiler, translate the program into a machine language.

5. Test and debug the program.

6. Put the program into production.

7. Maintain and enhance the program.

* …(5)

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **2** | **What Is Pseudocode?** | **5** |

Flowcharts and pseudocode are two common formats for displaying algorithms. Since pseudocode is simple to read and write and enables the programmer to concentrate on the logic of the problem, it has been chosen as the primary technique of encoding an algorithm. English is structured in pseudocode. To resemble the high-level computer languages, English has been formalized and shortened. Program logic can also be represented using pseudocode. The strategy is to write the program in a language that is similar to the implementation language while using understandable shorthand for difficult statements. The goal is to describe the logic while ignoring many of the programming language's syntactic restrictions. Thus, a pseudocode design description combines human language with coding elements. Simple statements are used in place of the full reasoning for well-known functions. Similar to how conditional statements are used to depict program flow, written expressions can be used to convey the logic. It is a good idea to employ language constructs that are comparable to those in the high-level language being used, even if there are no widely agreed standards for pseudocode notation. When you do, the pseudocode will appear familiar during implementation and will offer a framework. Pseudocode is typically simpler and quicker to generate than a finished source program because it does not involve knowledge of the specifics of the programming language. Although there is no universal pseudocode, once you comprehend one form, it is simple to comprehend others. A lot of programmers employ pseudocode. It is used to express abstract algorithms in their abstract forms. It is expressed in more language-specific forms to express design for programs written in those languages. Pseudocodes are designed to be simple enough for a human to execute "by hand," without the aid of a computer and with little to no programming experience. The algorithm is represented using pseudocode.

* …(5)

|  |
| --- |
| **SESSION 2.**  **Describe different data representations used in computer programs.** |
| **Learning Outcomes** |
| * The description distinguishes between different numeric data types (at least 3). * The description identifies different logical data types. * The description distinguishes between different internal representations of data types. * The description identifies different logical operators. |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **3** | **Identify different logical data types.** | **8** |

**Numeric data**

Numeric data simply means numbers.

**Integers**

An integer is a whole number - it has no decimal or fraction parts.

**Real Number**

Any number that you could place on a number line is a real number. Real numbers include whole numbers (integers) and numbers with decimal/fractional parts. Real numbers can be positive or negative.

**Currency**

Currency refers to real numbers that are formatted in a specific way. Usually, currency is shown with a currency symbol and (usually) two decimal places.

**Percentage**

Percentage refers to fractional real numbers that are formatted in a specific way - out of 100, with a percentage symbol.

**Alphanumeric (Text) Data**

Alphanumeric (often simply called 'text') data refers to data made up of letters (alphabet) and numbers (numeric). Usually symbols ($%^+@, etc.) and spaces are also allowed.

**Date and Time data**

Date (and time) data is usually formatted in a specific way. The format depends upon the setup of the computer, the software in use and the user’s preferences.

**Boolean (Logical) Data**

Boolean data is sometimes called 'logical' data (or in some software, 'yes/no' data). Boolean data can only have two values: TRUE or FALSE

* …(8)

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **4** | **Identify and describe different logical operators.** | **7** |

**Logical operators** are symbols or words used in logic and computer programming to perform logical operations on one or more Boolean values (true or false).

1. **NOT Operator (!):**

Takes a single Boolean operand and returns the opposite value.

For an example If the input is true, it returns false, and if the input is false, it returns true.

**Application:** “!true” returns false.

1. **OR Operator (||):**

Requires two Boolean operands.

Returns true if at least one of the operands is true.

Returns false only if both operands are false.

1. **AND Operator (&&):**

Requires two Boolean operands.

Returns true only if both operands are true.

If one or both operands are false, it returns false.

* …(6)

|  |
| --- |
| **SESSION 3.**  **Describe the basic principles of Computer Programming.** |
| **Learning Outcomes** |
| * 1. The description identifies different algorithmic structures of programming languages. * 2. The description identifies good program documentation principles (at least 3). * 3. The description identifies programming quality assurance (QA) principles. * 4. The description distinguishes between validation and verification. * 5. The description explains the relationship between files, records and fields. |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **5** | **Identify and describe different algorithmic structures of programming languages** | **8** |

Sequential Structure:

Description: The sequential structure is the most basic algorithmic structure, where statements are executed in a linear, sequential order, one after another.

Use: It is used for straightforward, step-by-step execution of instructions.

Conditional Structure (Selection):

Description: Conditional structures allow you to make decisions in your code based on conditions. Common forms include "if-else" and "switch" statements.

Use: Conditional structures are used to create branching logic, executing different code blocks based on whether specific conditions are met.

Looping Structure (Iteration):

Description: Looping structures allow you to execute a block of code repeatedly while a certain condition is true. Common loops include "for," "while" and "do-while" loops.

Use: Looping structures are used for tasks that require repetitive execution, like processing arrays or performing calculations until a condition is met.

Recursive Structure:

Description: Recursive algorithms involve a function calling itself to solve a problem. It breaks a problem into smaller, similar subproblems until a base case is reached.

Use: Recursion is often used for solving complex problems that can be divided into smaller instances of the same problem.

Parallel Structure (Concurrency):

Description: Parallel structures enable the execution of multiple tasks simultaneously, either using threads or processes. Languages with built-in support for concurrency provide constructs like "threads" or "coroutines."

Use: Parallel structures are used for tasks that can be executed concurrently, such as multi-threaded applications or distributed systems.

* …(6)

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **6** | **Identify and describe programming quality assurance (QA) principles.** | **8** |

Requirements Validation: Before starting development, ensure that the project requirements are clear, complete, and well-defined. QA should verify that the requirements are testable and meet the needs of all stakeholders.

Early Testing: Begin testing as early as possible in the software development life cycle. This includes unit testing, integration testing, and other forms of testing to catch defects before they propagate through the system.

Code Reviews: Implement a code review process where developers review each other's code for correctness, readability, and adherence to coding standards. Code reviews help identify and fix issues early. Automated Testing: Develop and maintain a suite of automated tests, including unit tests, integration tests, and regression tests. Automated testing reduces manual effort and ensures that changes do not introduce new bugs. Continuous Integration (CI) and Continuous Deployment (CD): Set up CI/CD pipelines to automate the build, testing, and deployment processes. This ensures that code changes are regularly integrated, tested, and deployed to production, reducing the risk of integration problems.

* …(8)

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **7** | **Distinguishes between validation and verification** | **4** |

Verification:

Verification is the process of checking whether a product, system, or component meets specified requirements and standards. It involves reviewing documents, conducting inspections, and performing various tests to ensure that the product or system is designed and built correctly. Verification answers the question, “Are we building the product right?”

Validation:

Validation is the process of evaluating a product, system, or component during or at the end of the development process to determine if it meets the customer’s or user’s needs and expectations. It typically involves testing the product with real users or in a real-world environment to ensure it functions as intended and provides value to the users. Validation answers the question, "Are we building the right product.

* …(4)

|  |
| --- |
| **SESSION 4.**  **Describe the principles used in designing a computer program.** |
| **Learning Outcomes** |
| * The description identifies methods of specifying problems. * The description explains techniques used to research problems in terms of inputs and outputs. * The description includes an evaluation of the viability of developing computer programs to solve problems and it identifies the issues in assessing the viability. * The description explains the features of a computer program that could solve a given problem. |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **8** | **Identify methods of specifying problems in designing a computer program.** | **4** |

**Use Cases:** Use cases are narratives that describe how different actors (users or systems) interact with the program to achieve specific goals. They help in identifying functional requirements and understanding user interactions.

**Problem Statement:** Begin by writing a clear and concise problem statement that describes the issue the program is intended to solve. This statement should outline the problem's context, its scope, and the desired outcomes.

**Pseudocode:** Pseudocode is an informal, high-level description of the program's logic, written in plain language. It helps in breaking down the problem into smaller steps and can serve as a.

**Flowcharts:** Flowcharts are visual representations that show the flow of control and data within the program. They can help specify the logical structure of the software, including decision points and loops.

* …(4)

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **9** | **Identify features of a computer program that could solve a given problem.** | **7** |

**Input Handling:**

Determine how the program will receive and process input data. This may involve user input through a graphical user interface (GUI), command-line arguments, file reading, or data retrieved from external sources like databases or APIs.

**Data Storage:**

Identify the data structures and databases needed to store and manage data efficiently. Choose the appropriate data types and storage mechanisms based on the nature of the problem and data volume.

**Algorithms and Logic:**

Define the algorithms and logic required to solve the problem. Consider algorithmic complexity and efficiency when selecting appropriate algorithms for data manipulation, sorting, searching, and other operations.

**Error Handling and Validation:**

Implement error-handling mechanisms to handle unexpected situations gracefully. Validate input data and provide informative error messages to users.

User Interface (UI):

Determine if the program requires a user interface and design it accordingly. GUIs are essential for user interaction, while command-line interfaces (CLI) may be suitable for certain applications.

* …(6)

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **10** | **Demonstrate understanding of carrying out an evaluation of the viability of developing computer programs to solve problems and identifies the issues in assessing the viability.** | **5** |

A crucial element in the software development process is assessing the possibility of creating computer programs to address issues. This assessment aids in deciding whether it is practicable, useful, and worthwhile to move forward with the development. Understanding how to do this evaluation and the problems that can occur are provided below:

**Feasibility Study:**

Conduct a feasibility study to assess whether it's technically, economically, and operationally feasible to develop the software.

**Technical Evaluation:**

Determine the technical feasibility of implementing the required features and functionalities.

**Cost Analysis:**

Estimate the overall development and maintenance costs, including hardware, software, personnel, and ongoing support.

**Problem Definition and Understanding:**

Begin by clearly defining and understanding the problem that the computer program is intended to solve. Ensure that the problem is well-defined and that there is a genuine need for a software solution.

**Legal and Regulatory Compliance:**

Assess whether the development and deployment of the software would comply with legal and regulatory requirements, such as data privacy laws, industry standards, and intellectual property rights.

* …(5)