Unit 1: Principles of Computer Science

Level: **3**

Unit type: **External**

Guided learning hours: **120**

Unit in brief

This unit covers the principles that underpin all areas of computer science. It will develop your computational-thinking skills and you will apply those skills to solve problems.

Unit introduction

Problem solving is an essential skill in all areas of life. To be successful, professionals need to be able to analyse the needs of individuals and organisations, and to evaluate the suitability and effectiveness of current ways of working in order to develop solutions that improve or enhance processes and/or outcomes.

In this unit, you will explore the logical and structured ways that computer systems process data to develop programs, processes and systems that solve specific problems. You will examine the features of effective computer programming and apply accepted computing and programming paradigms. You will analyse, develop and evaluate algorithms and computer code, and propose and apply solutions to ensure that computer systems are fit for purpose. To complete the assessment task within this unit, you will need to draw on your learning from across your programme.

In this unit, you will develop the computational-thinking skills to effectively analyse a problem, break it down into its component parts, and design and evaluate solutions. These skills are required for progression to computing-related higher education courses or to the workplace as a computing professional.

Summary of assessment

This unit is assessed through a written examination set and marked by Pearson.

The examination is two hours in length. During the supervised assessment period, learners will be assessed on their ability to apply their computational-thinking skills to solve problems.

The number of marks for the unit is 90.

The assessment availability is January and May/June each year. The first assessment availability is May/June 2017.

Sample assessment materials will be available to help centres prepare learners for assessment.

Assessment outcomes

AO1 Demonstrate knowledge and understanding of computing facts, terms, standards, concepts and processes

Command words: complete, draw, give, identify, name, state

Marks: ranges from 1 to 5 marks

AO2 Apply knowledge and understanding to communicate understanding of computing facts, terms, standards, concepts and processes

Command words: calculate, complete, demonstrate, describe, draw, explain, produce

Marks: ranges from 1 to 5 marks

AO3 Select and use computing technologies and procedures to explore outcomes and find solutions to problems in context

Command words: calculate, demonstrate, develop, explain, produce

Marks: ranges from 1 to 6 marks

AO4 Analyse data and information related to computer science in order to predict outcomes and present solutions

Command words: analyse, demonstrate, discuss, produce, write

Marks: ranges from 6 to 12 marks

AO5 Evaluate technologies, procedures, outcomes and solutions to make reasoned judgements and make decisions

Command words: evaluate, produce, write

Marks: ranges from 6 to 12 marks

Essential content

The essential content is set out under content areas. Learners must cover all specified content before the assessment.

A Computational thinking

Application of the thinking skills involved in analysing problems and processes, to identify solutions that can be developed into computer programs.

A1 Decomposition

- Identifying and describing problems and processes.
- Breaking down problems and processes into distinct steps.
- Describing problems and processes as a set of structured steps.
- Communicating the key features of problems and processes to others.

A2 Pattern recognition

- Identifying common elements or features in problems or systems.
- Identifying and interpreting common differences between processes or problems.
- Identifying individual elements within problems.
- Describing patterns that have been identified.
- Making predictions based on identified patterns.

A3 Pattern generalisation and abstraction

- Identifying information that is necessary to solve an identified problem.
- Filtering out information that is not needed to solve an identified problem.
- Representing parts of a problem or system in general terms by identifying:
  - variables
  - constants
  - key processes
  - repeated processes
  - inputs
  - outputs.

A4 Algorithm design

- Describing a step-by-step strategy to solve a problem.

B Standard methods and techniques used to develop algorithms

Techniques used to design solutions to problems.

B1 Structured English (pseudocode)

Produce, apply and interpret pseudocode statements to describe computing tasks or processes and solve problems.

- Interpreting pseudocode:
  - apply processes to calculate outcomes
  - evaluate the structure and logic of given code against given requirements
  - suggest improvements to logical structures and processes.
- Developing pseudocode:
  - improve the effectiveness and efficiency of code
  - identify and fix errors within code.

- Producing pseudocode – learners must be familiar with the listed terms and their application. Unfamiliar pseudocode will be given with definitions for application in the examination context:
  - sequence
  - structure:
    - hierarchy
    - indentation
  - operations:
    - BEGIN
    - END
    - INPUT
    - OUTPUT
    - PRINT
    - READ
    - WRITE
  - decisions:
    - IF
    - THEN
    - ELSE
    - ELSEIF (ELIF)
    - WHEN
  - repetition:
    - FOR
    - REPEAT UNTIL
    - WHILE
    - WHILE NOT.

B2 Flowcharts using standard symbols

Interpret, produce and develop flowcharts using appropriate British Computer Society (BCS) symbols to describe a system or solution.

- Process.
- Decisions.
- Input/output.
- Connectors.
- Start/end.

C Programming paradigms

Use of standard structures and conventions to build and develop accurate, efficient and effective computer code to fulfil identified criteria and solve problems.

C1 Handling data within a program

Selecting, applying, using and interpreting common data-handling techniques and structures provided within programming languages to process data.

- Defining and declaring constants and variables:
  - alphanumeric strings
  - arrays
  - Boolean
  - characters
  - date/time
  - floating point (real)
  - integers
  - objects
  - records
  - sets
  - strings.

- Managing variables:
  - local and global variables
  - naming conventions.

C2 Arithmetic operations

Selecting, applying, using and interpreting general mathematical expressions within computing structures to process data.

- Mathematical operators:
  - +
  - –
  - / (DIV)
  - *
  - %/MOD/modulo/rem.
- Relational operators (=, <, >, <>, <=, >=).
- Boolean operators (NOT, AND, OR).
- Date/time.

C3 Built-in functions

Selecting, applying, using and interpreting common functions provided within programming languages to perform specific tasks to process data.

- Arithmetic functions:
  - random
  - range
  - round
  - truncation.
- String handling functions:
  - concatenation
  - length
  - position
  - string conversion:
    - integer/float to string
    - string to integer/float.
- General functions:
  - input
  - open
  - print
  - range.

C4 Validating data

Selecting, applying, using and interpreting validation techniques to analyse and improve the accuracy and validity of data.

- Validation check techniques:
  - data type
  - range
  - constraints
  - Boolean.
- Post-check actions.

C5 Control structures

Selecting, applying, using and interpreting common programming control structures to analyse and improve the effectiveness of code.

- Loops:
  - REPEAT
  - FOR
  - WHILE
  - BREAK.
- Branches:
  - IF
  - THEN
  - ELSE
  - ELSEIF (ELIF).
- Function calls:
  - defining functions
  - declaring arguments
  - calling functions.

C6 Data structures

Selecting, applying, using and interpreting common data structures within a computer program to store and process data.

- Lists.
- Arrays:
  - single dimensional arrays
  - multi-dimensional arrays.
- Records.
- Sets.

C7 Common/standard algorithms

Selecting, applying, using and interpreting standard algorithms within a computer program to store and process data.

- Sorting:
  - bubble sort
  - quick sort
  - insertion sort.
- Searching:
  - serial/linear search
  - binary search.
- Other standard algorithms:
  - count occurrences
  - input validation.
- Using stacks and queues to implement sorting and searching:
  - Last In First Out (LIFO)
  - First In First Out (FIFO).

D Types of programming and mark-up languages

The features, applications, impact and implications of using different programming paradigms to develop code to solve problems.

D1 Procedural programming

Interpret, analyse and evaluate the features and applications of procedural programming paradigms in terms of:

- structure:
  - statements
  - blocks
  - procedures
  - functions/sub-routines
- control structures:
  - sequence
  - conditional
  - iterative.

D2 Object-orientated programming

Interpret, analyse and evaluate the features and applications of object-orientated programming in terms of:

- structure:
  - classes
  - objects/instances
- features:
  - inheritance
  - encapsulation
  - polymorphism and overloading
  - data hiding
  - reusability.

D3 Event driven programming

Interpret, analyse and evaluate the features and applications of event driven programming paradigms in terms of:

- structure:
  - main loop
  - callback function
  - sub-routines
- features:
  - events
  - event handlers
  - event loops
  - service orientated processing
  - time driven
  - trigger functions.

D4 Coding for the web

- The issues and implications of implementing code on a web platform in terms of:
  - performance
  - platform independence
  - power
  - protocols and APIs (Application Programming Interface)
  - security.

- The uses, applications and implications of client side processing and scripting.
- The uses, applications and implications of server side processing and scripting.

D5 Translation

The issues and implications of translating code between programming languages including:

- reasons for translating code from one language to another
- benefits of translating code from one language to another
- drawbacks of translating code from one language to another
- the implications of translating code and the impact on:
  - users
  - organisations
  - developers
- alternative ways to implement current code base.

Grade descriptors

To achieve a grade a learner is expected to demonstrate these attributes across the essential content of the unit. The principle of best fit will apply in awarding grades.

Level 3 Pass

Learners are able to use problem-solving skills to develop a solution to given problems in context. Learners use standard programming constructs to demonstrate an understanding of how data is handled in a computer program. Learners are able to construct, propose, develop and explain solutions to a problem and demonstrate an understanding of data validation and error checking.

Level 3 Distinction

Learners are able to analyse and interpret given problems and develop a detailed and complex solution in response. Learners demonstrate an in-depth understanding of programming constructs and a thorough understanding of how data is handled in a computer program.

Key terms typically used in assessment

The following table shows the key terms that will be used consistently by Pearson in our assessments to ensure students are rewarded for demonstrating the necessary skills.

Please note: the list below will not necessarily be used in every paper/session and is provided for guidance only.

Command or term	Definition
Analyse	Learners examine in detail, a scenario or problem to discover its meaning or essential features. Learners will break down the problem into its parts and show how they interrelate. There is no requirement for any conclusion.
Calculate	Learners apply some form of mathematical or computational process.
Complete	Learners complete a diagram or process. Can apply to problems/solutions of varying complexity.
Demonstrate	Learners illustrate and explain how an identified computer system or process functions. May take the form of an extended writing response, a diagram or a combination of the two.
Describe	Learners provide an account of something, or highlight a number of key features of a given topic. May also be used in relation to the stages of a process.
Develop	Learners provide a solution to a problem, typically using an existing system or structure that must be improved or refined.
Discuss	Learners investigate a problem or scenario, showing reasoning or argument.

Command or term	Definition
Draw	Learners represent understanding through the use of a diagram or flowchart.
Evaluate	Learners review and synthesise information to provide a supported judgement about the topic or problem. Typically a conclusion will be required.
Explain	Learners make a series of linked points and/or justify or expand on an identified point.
Identify	Learners assess factual information, typically when making use of given stimuli. Requires a single word or short sentence answer.
Produce	Learners provide a solution that applies established constructs to a given computing problem.
State, name, give	Learners assess factual information. Requires a single word or short sentence answer.
Write	Learners produce a solution, or a mechanism used as part of a solution, to a given computing problem.

Links to other units

This assessment for this unit should draw on knowledge, understanding and skills developed from:

- Unit 2: Fundamentals of Computer Systems
- Unit 3: Planning and Management of Computing Projects
- Unit 4: Software Design and Development Project
- Unit 5: Building Computer Systems
- Unit 6: IT Systems Security
- Unit 7: IT Systems Security and Encryption
- Unit 8: Business Applications of Social
- Unit 9: The Impact of Computing
- Unit 10: Human-computer Interaction
- Unit 18: Relational Database Development
- Unit 20: Managing and Supporting Systems
- Unit 21: Virtualisation
- Unit 22: Systems Analysis and Design
- Unit 23: Systems Methodology
- Unit 24: Software Development
- Unit 26: Programmable Devices and Controllers
- Unit 28: Computer Forensics.

Employer involvement

Centres may involve employers in the delivery of this unit if there are local opportunities. There is no specific guidance related to this unit.