Centre Guidance for Skills Mapping Summer 2022

Pearson Edexcel Level 1/Level 2 GCSE (9–1) in Computer Science Paper 2 – Application of Computational Thinking (1CP2/02)

Introduction

As part of the ResultsPlus service, teachers and centres have access to tools to help analyse student performance. It is not possible to use this same service for the 1CP2/02, the Python programming paper.

This document provides teachers and centres with the data required to carry out an analysis, similar to that from ResultsPlus, for 1CP2/02.

Skills map

The skills mapping shown in the tables show the questions from summer 2022 1CP2/02 mapped to the specification point.

Interpreting the skills map

The broad nature of the descriptions in the specification allows for some interpretation about where demonstration of a specific skill could be mapped. For example, use of the print() subprogram, could be mapped to 6.6.1, but may also be mapped to 6.4.1.

The total marks for a question may be made up from any number of specification points. Therefore, some specification points may earn more than one mark per question. A 10-mark question may cover exactly ten individual specification points.

Recall that marks in the levels-based mark schemes are awarded on a best-fit model. They use a holistic approach. The considerations provided for their award do not have to be fully met. Nor, on the other hand, are the considerations a limitation of what those marks can be awarded for. For example, there may be five potential specification points that can be addressed with only the three marks available in the Design LBMS.

Skills map tables

The tables below show the questions from summer 2022 1CP2/02 mapped to the specification point, along with a description of that point.

Q01

Subject content	Mark point	Students should:
6.3.1	1.1	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and oneand two-dimensional structured data types (string, array, record)
6.3.2	1.8	be able to write programs that make appropriate use of variables and constants
6.3.3	1.9	be able to write programs that manipulate strings (length, position, substrings, case conversion)
6.4.1	1.2 1.10	be able to write programs that accept and respond appropriately to user input
6.4.3	1.5	understand the need for and be able to write programs that implement validation (length check, presence check, range check, pattern check)
6.5.1	1.7	be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)
6.5.2	1.4	be able to write programs that use relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)
6.5.3	1.6	be able to write programs that use logical operators (AND, OR, NOT)
6.6.1	1.3	be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions)

Q02

Subject content	Mark point	Students should:
6.1.2	2.6	be able to read, write, analyse and refine programs written in a high-level programming language
6.1.3	2.9	be able to convert algorithms (flowcharts, pseudocode*) into programs
6.1.5	2.2 2.3 2.4 2.5	be able to identify, locate and correct program errors (logic, syntax, runtime)
6.1.6	2.7	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)
6.3.1	2.1	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one- and two-dimensional structured data types (string, array, record)
6.3.2	2.8	be able to write programs that make appropriate use of variables and constants
6.1.2	2.10	able to read, write, analyse and refine programs written in a high-level programming language

Q03

Subject content	Mark point	Students should:
6.1.2	3.8	Be able to read, write, analyse and refine programs writtin in a high-level programming language
6.1.3	3.3	Be able to convert algorithms (flowcharts, pseudocode*) into programs
6.1.6	3.9	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)
6.3.1	3.2	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and oneand two-dimensional structured data types (string, array, record)

6.5.1	3.5 3.6 3.7	be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)
6.5.2	3.4	be able to write programs that use relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)
6.6.1	3.1 3.10	be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions)

Q04

Subject content	Mark point	Students should:
6.1.1	4.1 4.2	be able to use decomposition and abstraction to analyse, understand and solve problems
6.1.6	4.14 4.15	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)
6.2.2	4.3 4.10	be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled, condition-controlled), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms
6.3.1	4.5	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and oneand two-dimensional structured data types (string, array, record)
6.3.3	4.4	be able to write programs that manipulate strings (length, position, substrings, case conversion)
6.4.1	4.9 4.12 4.13	be able to write programs that accept and respond appropriately to user input
6.5.1	4.6 4.7	be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)
6.6.1	4.11	be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions)
6.6.2	4.8	be able to write functions that may or may not take parameters but must return values, and procedures that may or may not take parameters but do not return values

Q05

Subject content	Mark point	Students should:
6.1.2	5.13	be able to read, write, analyse and refine programs written in a high-level programming language
6.1.4	5.9	be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain
6.1.6	5.14	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)
6.2.2	5.4 5.10	be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled, condition-controlled), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms
6.3.1	5.3	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one- and two-dimensional structured data types (string, array, record)
6.3.2	5.2 5.5	be able to write programs that make appropriate use of variables and constants
6.3.3	5.7 5.11 5.12 5.15	be able to write programs that manipulate strings (length, position, substrings, case conversion)
6.4.2	5.1 5.6 5.8	be able to write programs that read from and write to comma separated value text files

Q06

Subject content	Mark point	Students should:
6.1.1	6.7	Develop Code, be able to use decomposition and abstraction to analyse, understand and solve problems
6.1.2	6.10	Develop Code, be able to read, write, analyse and refine programs written in a high-level programming language
6.1.4	6.11 6.12	Develop Code, be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain

6.1.6	6.14 6.15	Develop Code, be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)
6.2.2	6.8	Constructs, be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled, condition-controlled), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms
6.3.1	6.3 6.4 6.6 6.9	Data Structures, be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and oneand two-dimensional structured data types (string, array, record)
6.3.2	6.5	Data Structures, be able to write programs that make appropriate use of variables and constants
6.3.3	6.1	Data Structures, be able to write programs that manipulate strings (length, position, substrings, case conversion)
6.4.1	6.13	Input and Output, be able to write programs that accept and respond appropriately to user input
6.6.1	6.2	Subprograms, be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions)

Using the skills map

To determine the performance of an individual student against the specification points, carry out these steps:

1. Mark a student response for a question
2. Note the mark point awarded
3. Map the mark point awarded to a specification point, using the tables above

To determine the performance of a cohort, aggregate the data.

Further support

Additional documentation about this qualification can be found on the Pearson Edexcel subject-specific page.

On that same page, you will be able to find ways to contact the subject advisor.