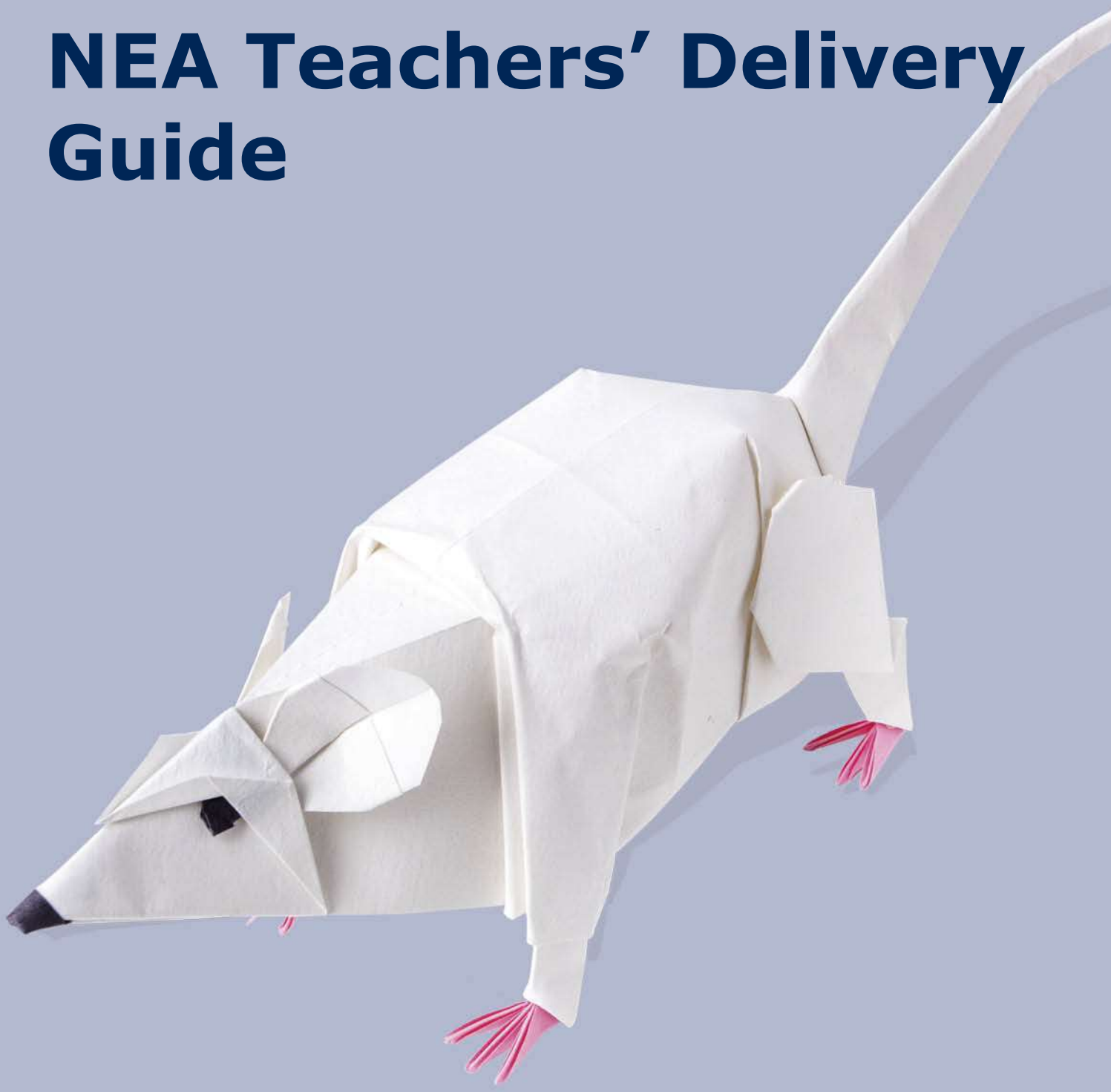
edexcel

NEA Teachers' Delivery Guide



GCSE (9-1) Computer Science

Pearson Edexcel Level 1/Level 2 GCSE (9-1) in Computer Science (1CP1)

ALWAYS LEARNING

PEARSON

Contents:

Purpose of this document

The purpose of this document is to help you prepare your students to complete the Edexcel GCSE (9-1) Computer Science non-examination assessment (NEA). It gives practical advice as well as guidance on the required regulations to follow whilst taking and marking the assessment.

Information on the NEA Project is on **pages 14 to 29** in the Edexcel GCSE (9-1) Computer Science specification. This guide is not intended as a replacement to the information already published but provides supporting guidance to assist teaching and learning.

Requirements for the NEA

- The weighting of the NEA is 20% of the qualification
- The maximum number of hours to complete the work is 20 hours
- The programming language used is a high-level programming language that has a textual program definition

Key dates for the NEA

The revised dates for the GCSE Computer Science NEA are:

- Tasks will be released in September prior to the final assessments. For example, tasks will be released on 1 September 2017, for students taking their exams in summer 2018.
- The deadline for submission of NEA marks will be **31 March**, rather than 15 May. This is to allow awarding organisations time to conduct additional checks during moderation.

Overview

The purpose of this component, which takes the form of a project, is to test student skills in responding to computer science problems.

We will provide a project brief that describes a problem that students will need to solve by developing a computer program.

If a data file is required to complete the project this will be provided by Pearson.

Students will submit their program with a written report showing how the program was developed, tested and evaluated. As in most systems, the set project task will require students to create a program that will include the following:

- data input and storage
- processing data
- producing output based on processed data

There are four different stages in the project, these are:

1. Analysis
2. Design
3. Implementation
4. Testing, refining and evaluation

Programming languages

There are five potential language choices for Paper 3 1CP1/3A-3E each with a different unit code based on the language chosen. These are:

- Python (1CP1/3A)
- Java (1CP1/3B)
- C-derived (C, C++, C#) (1CP1/3C)
- Visual Basic.NET (1CP1/3D)
- Pascal/Object Pascal (1CP1/3E)

Scheduling the NEA

Schemes of Work

The assumption in this document is that students will complete the whole course over a period of two years, with two hours per week timetabled.

- In Year 10 students should start learning the theory topics in the specification and can consolidate/learn programming skills to enable them to complete the project in Year 11.
- In Year 11 the theory should continue and a practice NEA can be completed in the Autumn Term. This will prepare students to start the current Project Brief towards the end of the Autumn Term. The NEA will be completed in the Spring term, which will then be followed by revision lessons.

A suggested Course Planner and detailed Schemes of Work for a two-year programme are available here:

http://qualifications.pearson.com/en/qualifications/edexcel-gcses/computer-science-2016.coursematerials.html#filterQuery=category:Pearson-UK:Category%2FTeaching-and-learning-materials

Teachers are of course free to use their own planners and schemes of work to adapt to fit their requirements and include a Year 9 option if necessary.

The NEA overlaps with 1CPO/02 which focuses on Topic 1: Problem Solving and Topic 2: Programming which the course planner suggests should be taught from the start of the course. This will reduce delivery time of the NEA.

Assessment scenarios may draw on Topics 2 to 6.

NEA preparation Scheme of Work example

This can be found in the Autumn year 11 and Spring Year 11 Schemes of Work support materials at https://qualifications.pearson.com/en/qualifications/edexcel-gcses/computer-science-2016.coursematerials.html#filterQuery=category:Pearson-UK:Category%2FTeaching-and-learning-materials&filterQuery=category:Pearson-UK:Document-Type%2FScheme-of-work

Total time allocated to working on the NEA

The maximum number of hours that students can spend on completing the NEA is 20. Teachers need to consider how to accommodate this within centre timetables. For example, if a class is scheduled for 1 hour, time is taken coming into the classroom, logging in and distributing materials. This could mean that the time spent in each lesson is maybe 45 to 50 minutes. Teachers might find it helpful to time this in a trial, and then build it into their NEA schedule. It is advisable to keep a log of the sessions.

Students who miss lessons are entitled to catch up time, and some students may have extra time allocated to them. This also needs to be timetabled.

Teachers can
Exclude time taken getting students ready to start working in each session from the 20 hours.
Allow extra time for students who are entitled to this, or who have missed lessons.

Recommended time for the different stages in the NEA

It is important that teachers keep track of their students' progress so that they do not run out of time to complete all the sections and thus lose marks.

Stage 1 Analysis (6 marks)

- It is recommended that no more than 2 hours is spent on Stage 1.

Stage 2 Design (18 marks)

- It is recommended that no more than 6 hours is spent on Stage 2.

Stage 3 Implementation (24 marks)

- It is recommended that no more than 10 hours is spent on Stage 3.

Stage 4 Testing, Refining and Evaluation (12 marks)

- It is recommended that no more than 2 hours is spent on stage 4.

The Secure Environment

Student accounts

Centres must set up separate user accounts for each student, which are only accessible by that student and their supervisor. These accounts must only be **accessible in the supervised environment**. It is recommended that centres install monitoring software so that they can monitor individual students as they are working on the NEA

Each account **must** hold:

- The programming language environment
- A data file for the current project brief (issued by Pearson if required)
- Word processing software
- Spreadsheet software, if required for the inspection of any text file provided

Each account **may** hold:

- An electronic version of the project brief
- An electronic version of a syntax help guide for the chosen language
- An electronic version of the pseudo-code guide issued by Pearson
- Graphics software for presentation of flowcharts.

Teachers can
Create an electronic syntax help guide, including basic syntax structure details.
Give the students a printed copy of the syntax help guide.
Give students a copy (simplified if required) of the Mark Scheme from the Specification.

Electronic access outside the student accounts

Teachers can	Teachers cannot
Allow students to use an Intranet if it is necessary to give access to the student accounts	Allow access to the Intranet for any other reasons than access to the student accounts
	Allow access to the Internet
	Allow students to bring in work to the environment electronically or remove work electronically (e.g. on USB or other portable storage media)

Materials in the supervised environment

Students can only work on their project in the supervised environment, which means that materials in any form cannot be removed or taken in to that environment. Teachers have a responsibility to explain the reasons for this to their students and have processes in place to make sure that malpractice does not occur.

Teachers can	Teachers cannot
Allow students to print off their work for use the supervised environment.	Allow students to bring in any materials (e.g. notes, reference materials) to the supervised environment.
Allow students to write on printed hard copies.	Allow students to remove the project brief from the supervised environment.
	Allow students to remove any other materials (e.g. notes, reference materials) from the supervised environment.

Additional resources

These are limited to:

- Pseudo code guide, from the specification (page 41)
- Flow chart symbols, from the specification
- Centre syntax guide (to be submitted with the work) **
- Testing template provided with the NEA
- Integrated Development Environment for the programming language selected
- Word processing software
- Spreadsheet software
- Graphics software for presentation of flowcharts.

** The syntax guide should provide a translation of the pseudo code in the specification into the programming language selected. It may be paper based, or provided as a digital resource.

Teachers MUST NOT
Provide any additional resources
Provide programming examples in any form

Storing materials securely

It is vital that students' work is stored securely. Students must store their ongoing work in their NEA user account areas.

Teachers MUST
Destroy any printed materials securely at the end of the session **OR** store securely for students to use in the next session.

Pre-compiled libraries, units or modules

Programming languages have standard libraries which enable programs to access standard functions, for example writing to files. However other custom-built libraries, units or modules are also available. If students wish to use custom libraries, units or modules they must inform their teacher who must approve them by following the guidelines in the specification on page 15.

Teachers can	Teachers cannot
Allow students to use standard libraries, units or modules in the chosen programming language.	Allow students to use custom libraries, units of modules without the teachers' approval.

Teachers MUST
Follow the guidelines on page 15 of the specification: use of pre-compiled libraries, units or modules.

Security and backups

It is the responsibility of the centre to keep secure the work that students have submitted for assessment.

Centres are strongly advised to utilise firewall protection and virus checking software and to employ an effective backup strategy, so that an up-to-date archive of students' evidence is maintained.

Guidance and feedback

Teachers can	Teachers cannot
Give generic class-wide feedback or instruction to help students understand rubrics, assessment criteria and controlled conditions. This should be of a general nature and be concerned with over-riding guidance for example in the Sample Assessment Material[1] an explanation of what a six-digit location reference is.	Provide templates, model answers or writing frames
Review candidates' work and provide oral and written advice at a general level, then having provided advice at a general level, allow candidates to revise and re-draft work.	Demonstrate an example of a solution to the actual problem, or a closely related problem, during controlled conditions or at any other time.
	Provisionally assess work and then allow the candidate to revise it.
	Use the actual project brief in practice sessions.

Collaboration

Students must be supervised and must not work with others to develop their project. Where computer workstations are situated close together, supervisors must ensure that students are working independently and not communicating with each other.

1 http://qualifications.pearson.com/content/dam/pdf/GCSE/Computer%20Science/2016/Specification%20and%20sample%20assessments/computer-science-sam.pdf

Malpractice

Teachers should familiarise themselves with the Joint Council for Qualifications Instructions and information for conducting non-examination assessments for the academic year in question. These can be found on the JCQ website http://www.jcq.org.uk/exams-office/non-examination-assessments.

They should also make sure students are aware of what constitutes malpractice and the consequences.

Any contravention of the guidelines in this document or the JCQ instructions will be considered as malpractice

Submission of work

Marking, standardisation and moderation

Key control procedures have been mentioned already and this section gives more detail.

The NEA is marked by centre staff. Where marking for this specification has been carried out by more than one teacher in a centre, there must be a process of internal standardisation carried out to ensure that there is a consistent application of the criteria laid down in the marking grids, across all the units. A **Non Examination Assessment Record** (NEAR) should completed for all students. This is available on the website. Teachers can annotate students work if they feel additional clarity is required.

Centre marks should be submitted via Edexcel Online by 31st March in the year of assessment.

Marks awarded by the centre will be subject to external moderation by Pearson. Following the submission of marks, Pearson will notify centres of the students whose responses have been selected for moderation.

If centres have a statistically unusual relationship between their written paper and NEA marks they will have additional, enhanced moderation checks.

In addition, centre visits will happen between September and March of the following academic year if:

- They had a statistically unusual distribution of written paper and NEA marks.
- They are suspected of malpractice in GCSE Computer Science (9-1) NEA (or GCSE Computing in the previous year).
- They have been found guilty of malpractice in GCSE Computer Science (9-1) NEA

Additionally, a small percentage of new and small centres will be visited.

Please refer to the JCQ Instructions for conducting Non Examined Assessments (GCSE qualifications) on the JCQ website: www.jcq.org.uk for further information.

Presentation

Students must present their work for the NEA electronically. Folders must be created by the centre according to the instructions given in the brief. Their submitted work will be:

- The written report
- The program source code and all associated files required to run the program.

Student files should be identified by student name and task and presented in a single folder. Students will be expected to present content in a format appropriate for viewing at a resolution of 1024 x 768 pixels.

Centres should then add the Non Examination Assessment Record for each of the students in the sample and submit all three files.

Sample work must be submitted in an approved digital format; that is, on CD-ROM or USB flash drives.

Delivering the project

Project stages detail

This section gives guidance on how to deliver each stage of the project. It should be read in conjunction with the GCSE (9-1) Computer Science specification from Edexcel. It also contains specific stage guidance on what support can be provided.

The contents of the written report are also included here as an aid to delivery. The written report should be thought of as a 'wrapper' to hold the work that will be assessed rather than a document where 'Quality of Communication'[2] used to be assessed. However clear and comprehensive writing may be relevant in the marking assessment criteria.

The new NEA requires students to make some decisions and assumptions of their own. They will have to work out a number of approaches by themselves as the project brief will not give them all the smallest details. The Edexcel NEA problem is an 'open' problem to encourage creativity when creating the solution, so there is no definitive solution that they will be assessed against.

An NEA Exemplar is available on the Pearson website.

Stage 1: Analysis, 6 Marks

Purpose

The purpose of the analysis stage is to identify the requirements of the problem, and what the proposed solution will do to meet the requirements.

The analysis tasks are to:

- analyse the given problem and identify the requirements of the program that will be designed, implemented and tested
- decompose the problem into manageable sub-problems, with an explanation of each.

[2] Quality of Communication is no longer assessed explicitly in the reformed GCSEs.

Report contents

The report will contain:

- a short introduction to the problem
- a list of the requirements of the problem that will be programmed
- decomposition of the problem into sub-problems, including
  - a short description of what each of the sub-problems will do
  - a short explanation of the reasoning behind the decomposition submitted

Delivering the Analysis

The requirements of the problem are key and are followed throughout the project from the analysis to the evaluation. By explaining the reason behind the decomposition students are applying computational thinking to the problem. During NEA preparation, encourage students to look at other problems, breaking them down into sub-problems to identify the requirements. They could use highlighters to identify these requirements, then practice putting them into succinct prose or bullet points which are measurable, with a short explanation about why they have decomposed like this. Sample Assessment materials and previous NEAs can be used for this.

For example:

> *I will create a menu to give the user 3 options which will then give more menu options.*
>
> 1. *Add data*
>   - *Customer data*
>   - *Product Data*
> 2. *Buy*
>   - *Choose what to buy and put in basket*
>   - *Checkout*
> 3. *Show account*
>   - *Display*
>   - *Edit*
>
> *I chose to split the menu into 3 main options with sub-menus rather than 9 main options because it will be easier to use by the customers.*

A visual representation is not a requirement but can be included and may gain marks.

You will need to make some assumptions and decisions of your own and assumptions should be stated at this stage.

There are only 6 marks for the Analysis, and the report asks for 'short' entries. This section is therefore unlikely to be more than two pages long.

Teachers can	Teachers cannot
Remind students of the work they completed for Analysis during the NEA preparation phase.	
Allow students to look at their previous work outside of the NEA lesson time and the supervised environment.	Allow students to use their previous work during the NEA lessons.

Different students have different skills and should not be penalised if they cannot complete enough work at this stage to enable them to progress. Help can be provided in this situation and accounted for in the mark given. In this situation:

Teachers can	Teachers cannot
Provide sufficient support to enable the learner to identify the requirements of the problem and/or data requirements so that they can carry on to the next stage of the project.	Provide support for describing sub–problems.
	Provide assistance to students who have produced an incomplete analysis that would not prevent them from moving on to the next stage.

Teachers **MUST**
Award marks accordingly, if support has been given. For example, if no work has been submitted then 0 must be awarded. If some attempt to identify the requirements has been done then 1 or 2 marks may be awarded.
Record any help given on the NEAR.

It is very important that teachers see the complete Analysis before students start on the next stage.

Awarding marks for analysis

To gain the top mark band students will have carefully analysed and decomposed the problem into a clear list of requirements. Sub-problems will have been identified which may link to each other and in a hierarchical structure. The requirements will all relate to the given problem and a logical explanation of why they chose to decompose in the way they did will be present.

Work assessed in the lowest mark band is likely to be disjointed and may have generic statements, e.g. '*calculations will be done*' as opposed to specific calculations in the requirements. There will be little evidence of decomposition.

Stage 2: Design, total 18 marks

Purpose

The purpose of the design stage is to describe what has to be done when implementing the solution and to suggest an appropriate strategy to test the solution.

There are two sub-stages:

2.1 Solution design

2.2 Test strategy and initial test plan

2.1 Solution design (12 marks)

The solution design task is to:

- Design an algorithm that meets the requirements of the problem using appropriate conventions (flowchart, pseudo-code).

Report contents

The report will contain:

- the algorithm(s)
- any refinements to the design identified during implementation, with reasons.

Delivering the Solution design

Students should depict the sub-problems they have identified in the Analysis in a flowchart, or pseudo-code, or a written description. They can use all three conventions if they want to.

Refinements are usually identified during implementation or testing and are assessed later in Stage 4. These refinements should be **added** to the original. This reflects 'real life' design and development when changes are required. It would be very unusual for a design to exactly mirror the final implementation.

An 'open' problem will be given which may result in many alternative, correct solutions. Students will be assessed on how and why they decomposed in the way they chose.

Teachers can	Teachers cannot
Show students what the algorithms should include (p21 of the specification)	
Remind students that decomposition is of the requirements identified in their Analysis.	
Encourage students to note their thoughts as they go through the decomposition process.	Suggest words that might be used to justify student's choice of decomposition.
Refer students to the pseudo-code booklet.	

Teachers **MUST**
Not encourage students to build the program and then reverse engineer the design. They may lose marks for refinements if they do.
Remind students that program code is not acceptable for algorithms in this stage.

Again, students should not be penalised if they cannot complete enough design work to progress further.

Teachers can	Teachers cannot
Provide sufficient support to enable the learner to develop a minimal solution that may not address all the requirements but will allow the student to develop a program.	Provide examples of algorithms (even generic ones).
	Provide assistance to students who have produced an incomplete design solution that would not prevent them from moving on to the next stage.

Teachers MUST
Award marks accordingly, if support has been given. For example, if some requirements have been identified and decomposed, and/or generic statements given, then a student could still gain 1-2 marks.
Record any help given on the NEAR.

Awarding marks for Solution design

To gain Level 3 marks a fully functional algorithmic solution will have been presented, addressing all the requirements and fully decomposed.

Lower marks are awarded if only limited attempts at decomposition, with little attempt to address the problem requirements have been made in the algorithmic solution. There will be significant logic errors and the student will not have used programming constructs appropriately. The solution will not function.

2.2 Test strategy and initial test plan (6 marks)

The task is to:

- Devise a test strategy based on meeting the requirements of the problem. The proposed strategy should be followed when creating an initial test plan and will be assessed from the contents. This must be completed before implementation and will be updated before the program is actually tested.

Test planning is done twice in the NEA, the first time is in this stage, and the second time in Stage 4. Each is assessed separately.

Report contents

The report will contain:

- the initial test plan, (which will follow the test strategy), using the headings shown

Test no	Purpose of the test	Test data	Expected result

When constructing test data for the initial test plan, normal data is data that the program will accept. Erroneous data is inaccurate data that the program will not accept. Boundary data is typically on the 'edge' of a range of possible values that may or may not be accepted. Not all tests may require data entry.

This is an initial test plan and more tests can be added later. At this stage the students should be thinking how to test their solution on how it meets the requirements they have identified.

During preparation work for the NEA and whilst learning to program students should be reminded that tests to cover normal, erroneous, and boundary data should be included where possible so that it becomes part of good practice.

It is very important that teachers see the Initial Test Plans before students begin to develop the program code.

Teachers can	**Teachers cannot**
Teach students the meaning of normal, erroneous and boundary data before the NEA commences.	Guide students directly to include these in feedback during the NEA completion.
Remind students that this test plan is based on the requirements identified.	
Explain the process of creating the Initial Test Plan before building the solution, then adding refinements to create the Final Test Plan in Stage 4.	

Teachers MUST
See the Initial Test Plan BEFORE students start to program the solution
Tell students that it should NOT be worked on once the students start developing the program code in Stage 4.
Ensure that the Initial Test Plan is saved and assessed separately from the Final Test Plan.

Students can be supported so that they are not prevented from carrying out some tests in Stage 4.

Teachers can	**Teachers cannot**
Provide some general indications of aspects of the program that need to be tested	Provide direct guidance to students to add tests that include normal, erroneous and boundary data.

Teachers MUST
Award marks accordingly, if support has been given. For example, a limited strategy not linked to requirements may be awarded 1-2 marks, but **one** of normal, erroneous or boundary data should have been included.
Record any help given on the NEAR

Awarding marks for the test strategy and initial test plan

Level 3 marks can be awarded for a comprehensive test strategy covering most if the problem requirements, and the test plan follows the strategy consistently.

Stage 3: Implementation, total 24 marks

Purpose

The purpose of the design stage is to program the solution to the problem.

There are two sub-stages:

3.1 Implementing the design

3.2 Building the solution

The two sub-stages in implementation (3.1 and 3.2) will happen concurrently as the solution develops. It is not possible to build a solution without implementing the design using programming concepts. Both sub-stages should, therefore, be marked at the same time using all the submitted code.

Report contents

The report will contain:

- A copy of the program code. Any refinements should be noted as comments in the final program and are assessed in Stage 4.
- screenshots demonstrating effective use of debugging skills to correct errors.

3.1 Implementing the design (6 marks)

This is where the algorithm(s) from the design are translated into the chosen high-level programming language. Students will apply the programming concepts from that language, e.g. they will apply the correct syntax of a pre-check loop.

Allowable support for this sub-stage:

Teachers can	Teachers cannot
Provide support by explaining syntax in general terms.	Give examples of how to implement, for example, program loops.
Provide a syntax guide (electronic or printed) on the chosen language.	Provide any programming code for the project.
	Help students to debug code.

3.2 Building the solution (18 marks)

The final programmed solution should address all the requirements listed in the analysis stage, and be functional. Computing techniques (comments, naming conventions, indentation) should be used and will be assessed.

Support is NOT allowed for this sub-stage.

Delivering the solution build

Developing programmed solutions is still computational thinking and this is the stage where it is most likely that gaps will be identified which can then added as refinements. These refinements should be implemented and documented as additions to the design (Stage 2) and in the program code by using comments as descriptors. A refinement can include many things, for example a more efficient way of programming a sub-problem or an additional component that has not been previously considered but which will add to the functionality of the solution.

Refinements will be awarded in Stage 4.

Students should be familiar with debugging techniques (hand-tracing, and the chosen language's debugging tools). By the time they get to the NEA debugging should be a normal part of programming to them and should be used when errors arise during this stage. Please note that this is not formal testing which will be assessed in Stage 4.

Evidence that they can and do use debugging is required here, with screenshots. Submit one or two screenshots showing debugging skills in actual use.

Awarding marks for Implementation

Although the marks for assessing the two sub-stages are separate, both sub-stages should be marked together since they will be completed concurrently.

3.1 Implementing the design gains marks depending on how well programming concepts have been applied.

3.2 Building the solution looks at the requirements, the completed solution, use of debugging tools, the use of data types and structures, and decomposition in programming.

The requirements are those identified in the analysis, and how well these have been addressed is assessed. This avoids discrimination against students who were unable to identify a comprehensive list of requirements. A limited or partial solution will still gain marks. Even the highest mark band allows for some omissions and errors, but with a fully functional program.

Stage 4: Testing, refining and evaluation, 12 Marks

Purpose

The purpose of this stage is to show that the final program solution has been tested along with any refinements, and the solutions evaluated against the original requirements.

Report contents

The report will contain:

- the updated and completed Test Plan (labelled and saved as 'Test Table')
- the evaluation

Delivering the Testing and refining and evaluation stage

Columns to show the 'Actual result' and 'Action needed/comments' should be added to the initial test plan. In this stage the actual tests are then carried out and these last two columns should be completed when each test is run. An example is shown. Any errors should have 'Action Needed/Comments' entries. An attempt should be made to correct and retest all errors.

Tests for any refinements completed in the design and/or implementation stages should be added to the end of the test plan and carried out.

Test no	Purpose of the test	Test data	Expected result	Actual result	Action needed/comments

Screenshots to evidence testing may be provided but are not required

Teachers should remember that they will have to authenticate students' work which will cover testing evidence.

A good test plan is likely to find errors. If a student does not find errors this is accommodated in the assessment criteria.

The evaluation should include a thorough and critical evaluation of the program. This should include how successfully the program meets each of the original requirements and the reason for adding refinements to the final solution.

The evaluation should **not** contain a log of what the students have done during the project.

Support is NOT allowed for this stage.

Awarding marks for testing, refining and evaluation

To gain the top mark band students will have tested all components and requirements, errors accounted for refinements added which improve functionality and program robustness. A critical review of how well the program meets the requirements is included.

Lower marks are awarded when only some of each of these criteria have been met.