A LEVEL COMPUTER SCIENCE

DISCLAIMER: *This presentation has been created before standardisation of the first cohort of H446 has taken place, and thus is NOT a precise definition of standards that should be applied. Guidance will be updated regularly. Please ensure you check the OCR website for the latest version of this presentation. The presentation is designed to support teachers in delivery of the H446 Component 03 – Programming Project.*

Programming Project Overview

- Allows candidate flexibility in choice of project
- Flexibility of language
  - Specified in Specification
  - Ask OCR if you want to use something different
- Now uses ‘Stakeholders’ rather than requirement for a ‘real’ 3rd party end user
- Push to focus on the development and coding sides
- Focus on ‘project complexity’
- Focus on RAD/Prototyping/Agile development
- No need for a User Guide
- Similar levels of support by teachers

Comparison of Old to New

Similar Features

- Same basic structure
- Similar marking areas
- Same wide choice of languages
- Same levels of teacher support
- Similar delivery time anticipated
- Will still interact with 3rd Parties
- Programming Techniques
- Graphical User Interface highly likely

Changes

- User Guide removed
- Level ICT in terms of complexity
- Updated mark scheme/banding
- Could start in Year 12
- Requirement for a ‘real end user’ removed to focus on Stakeholders
- Assessment of complexity is key
- Focus on Agile design

Where do I start?

- Prior Specification Sample Material
  - The H447 Sample Projects
  - Still strong ideas for current specification
  - Have a lot of good practice in
  - Would likely still do well within current assessment criteria
- Project Complexity Guide
  - Supports knowledge of ‘complexity’
  - Provides other sample ideas of projects

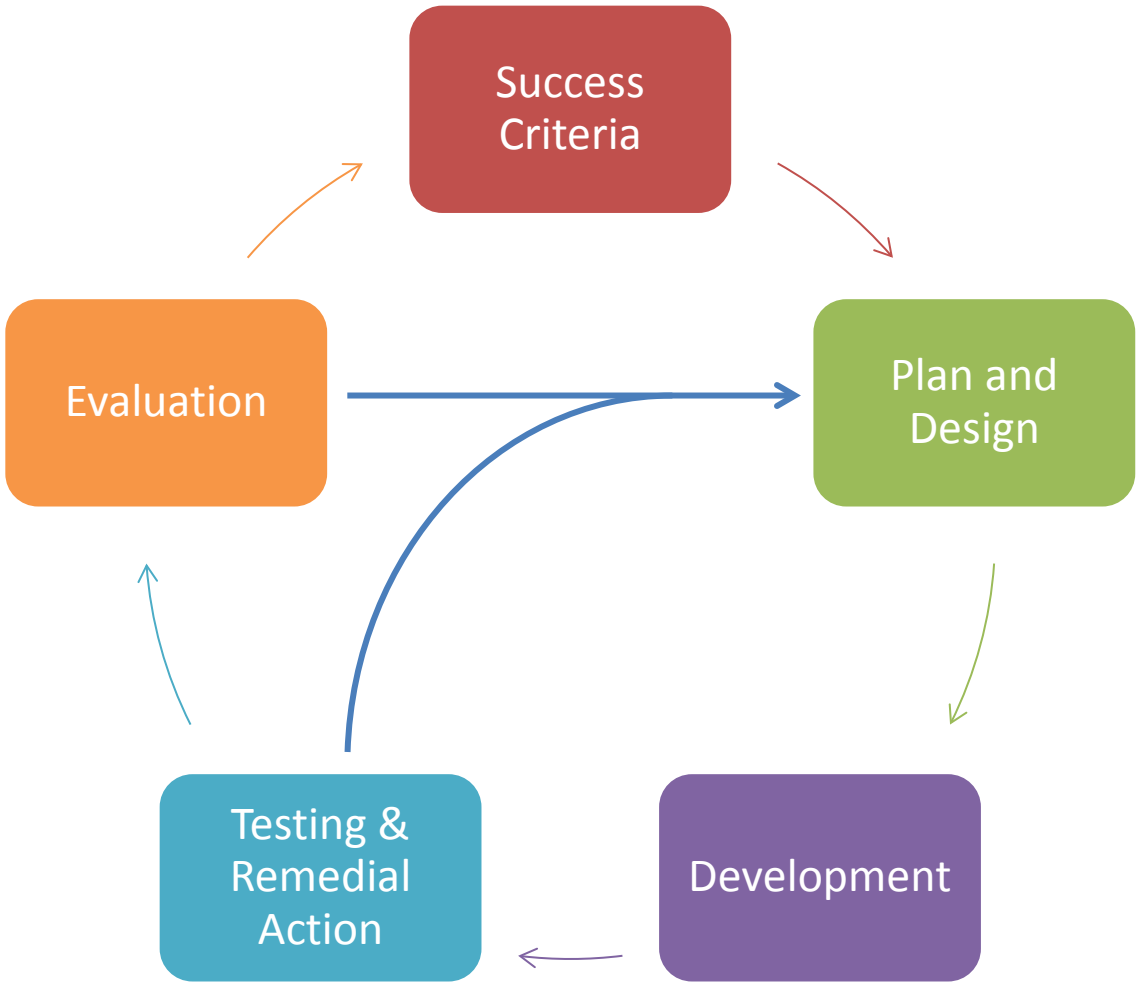
Where do I start?

- Read the mark scheme!
  - Note the changes in the requirements
  - Note changes in headings/sections
  - Be familiar with the expectations
  - Apply this to projects
- Discuss the mark scheme with learners to help them understand their targets

Preparation of Students

- Ensure they are familiar with the Process for Success
- Ensure they have learnt and understood all programming techniques/programming theory from Spec
- Ensure they are familiar with how they will be marked/assessed
- Ensure they pick a project they will engage with and be motivated to complete!

Iterative Development Process



Success criteria
- What will a successful solution do

Planning and design
- Pseudocode, flowcharts, DFDs, Class Definitions/UML etc

Development
- Narrative of steps taken
- Annotated Code
- Discussion of methodology

Testing and remedial actions
- narrative of changes made

Evaluation
- link to success criteria
- evidence of success or not

What languages can I use?

- Listed in the Specification
- We have also allowed use of:
  - Swift
  - JSON
  - NodeJS
  - Unreal/Unity Engines (linked to C# etc)
  - LUA
  - Robot X / Monkey X
  - JavaScript – expected to be used in conjunction with other things

What languages is ‘best’?

- Totally dependent on:
  - The project
  - Candidate Skills
  - Level of familiarity
- Programming Language Guide may help support choices
- Most pupils likely to have studied Python at GCSE if they opted for OCR (~90% of centres)
- There is no one ‘best’ language
- You may submit projects in a range of languages – there is no need to stick to one for all learners

How do I start?

- Brainstorm Ideas with students – in conjunction with the Complexity Guide
- Generate 10-15 ideas with each student
- Dot-vote the top 3
- Ask them to write a short proposal about each project
  - Include why they think it is a suitable project
  - Include what makes it complex
- Come up with the Winner

Acid Test

- Likely Complex Problems look like/may include the following ideas:

- Permanent Data Storage
- Potential for OO Paradigm
- Interfacing with hardware
- Exporting/Linking with other software
- Programs that learn/adapt over time
- Games/Physics
- Extended Logic Chains
- Use of libraries
- Use of A Level sorting/searching techniques
- Combination of differing technologies
- Sharing of data of LAN/WAN
- Expert Systems
- Simulators

What then?

- Please email us if you are concerned about any of the following:
  - Complexity
  - Project suitability
  - Approving Language Choice if not on spec

The Mark Scheme – brief overview

The following slides are not definitive in terms of guidance and direction.

Further guidance will be issued by the Principal Moderator for the unit in use of the mark scheme to assess the H447 Sample Work in due course.

Centres may always seek clarification be emailing us directly

Analysis – brief overview

9–10 marks
• Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research. • Identified the essential features of the proposed computational solution explaining these choices. • Identified and explained with justification any limitations of the proposed solution. • Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified and justified measurable success criteria for the proposed solution.

- Candidates will need to discuss why their project will be solvable using computational methods
- Clearly identify their stakeholders, and the roles they will play
- Show evidence of looking at other systems/ideas
- Link this research into their proposed designs
- Focus on key features (e.g. GUI Layout, Data structures) and explain whey they have chosen these
- Clearly delimit their problem and identify where they feel their project may face issues/limitations, discussing why these exist and the potential impact on the project
- Generate an overarching set of requirements for the project and also identify hardware/software requirements where needed:
  - This may link to Software Versions (e.g. Only working on certain releases of web browsers due to functionality requirements)
  - Hardware requirements – mostly focusing on unique hardware issues where presents. Generic discussion of a minimal set of requirements for a PC is not needed.
- Define a clear set of Success Criteria that is a summary of the above and where each one is justified in relation to the Stakeholders/research etc.

Design – brief overview

13–15 marks
• Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process. • Defined in detail the structure of the solution to be developed. • Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem. • Described, justifying choices made, the usability features to be included in the solution. • Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation. • Identified and justified the test data to be used during the iterative development of the solution. • Identified and justified any further data to be used in the post development phase.

- Candidates should be using a suitable design methodology for their chosen project + language. For instance, use of class diagrams for OOP, or Top Down Design for Procedural.
- Each smaller problem should be justified as to why this is a suitable "chunk" and how it fits in to the grand scheme
- Flow Charts, Data Flow, Pseudocode etc should be used appropriately to then design the program
- There is no specified design methodology
- Psuedocode is inherently required
- The emphasis is that a 3rd party should be able to implement the solution directly from the designs
- Suitable data structure design should be used as well
- Test data for the initial designs should be specified on a 'module by module' approach, as well as test data that will be used to ensure all of the Success Criteria are met
- Awarding of marks for Design and Test plans may be awarded at a later date if the candidate decides that initial designs/tests may need modification/adaptation

Development – brief overview

13–15 marks
• Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did and justifying why. • Provided evidence of prototype versions of their solution for each stage of the process. • The solution will be well structured and modular in nature. • Code will be annotated to aid future maintenance of the system. • All variables and structures will be appropriately named. • There will be evidence of validation for all key elements of the solution. • The development will show review at all key stages in the process.

- Avoid ‘death by screen shot’
- Prints of Code blocks/modules with descriptions are fine – no need for ‘line by line account’
- Highlight unique features or ‘complex coding’ etc – i.e. Showcase the complexity
- Develop in blocks/chunks as defined in the design
- Test as you go – no need to wait until the end
- Agile/Iterative/RAD styled development
- Good annotation in code will reduce quantity of writing
- ‘Lean and Mean’ – keep the development as concise as possible
- Ensure good naming conventions are used (should be included in designs). Candidates that use Form_1 etc, are not showing good maintainability of code.
- Highlight the validation – emphaise robust coding
- Check Points/Milestones after each module – review back to Success Criteria
- Signed of development by Stakeholders for each iteration

Testing (Development) – A Brief overview

9–10 marks
• Provided evidence of testing at each stage of the iterative development process. • Provided evidence of any failed tests and the remedial actions taken with full justification for any actions taken.

- Test at the end of each ‘module’
- No need to screenshot every test – especially if repetitious
- Highlight failed tests
- Must be enough to convince moderator of working solution
- It is fine to re-design etc at this point and you can still refer to this redesign to reflect a full working solution when awarding marks in the Design Section
- Any modification of test plans may also be used in the same way
- Must show re-development and testing where appropriate to gain full marks
- Ensure all white box testing (functionality) is completed at this stage.
- Testing for Use Acceptance (black box) is contained as part of evaluation
- This may form part of the testing they do as they develop, but is awarded in AO3.3, rather than AO3.2

Evaluation (Testing) – A Brief overview

5 marks
• Provided annotated evidence of post development testing for function and robustness. • Provided annotated evidence for usability testing.

- Black Box testing / User Acceptance Testing
- Involve all Stakeholders
- Identify issues raised from testing
- Again, it is fine to discuss solutions to any ‘failures’ and repeat the design/implementation phases to solve issues
- Must prove that the system meets all of the success criteria
- May reference successes from earlier in project – no need to repeat
- Reference clearly back to evidence through page numbers (and make sure they are correct!)

Evaluation – A Brief overview

13–15 marks
• Used the test evidence to cross reference with the success criteria to evaluate the solution explain how the evidence shows that the criteria has been fully, partially or not met in each case. • Provided comments on how any partially or unmet criteria could be addressed in further development. • Provided evidence of the usability features justifying their success, partial success or failure as effective usability features. • Provided comments on how any issues with partially or unmet usability features could be addressed in further development. • Considered maintenance issues and limitations of the solution. • Described how the program could be developed to deal with limitations of potential improvements / changes. • There is a well developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.

- Summary of final Success Criteria
  - Again – cross reference to any earlier evidence to save repetition
- Identify any potential limitations arising from the solution
  - A) They may go back and edit/customise if time allows
  - B) Critique if these are in fact future development problems
- Evidence that GUIs and Features are suitable and effective
- Discuss development of features that may be functional to meet success criteria, but could also be improved
- Explain impact of this development on final product
- Discuss how the system will need to be maintained
  - Refer to code maintenance
  - Evolving requirements of Stakeholders
- Refer to evidence, development, research etc. to support all of these justifications clearly.
- Ensure that sources are robust and wide-ranging

General Guidance

- Drive to reduce the volume of paperwork submitted
- Link back to evidence where needed – cut out repetition
- Focus on coding elements and complexity
- Reward design/testing plans where they occur and holistically for the mark band
  - (i.e. initial non-working designs that are then corrected will be rewarded)
- Removal of user guide – no requirement to have to write one any more
- Interaction with Stakeholders is fine – but not cumbersome!
- It may be appropriate to provide video evidence of working solution if write-up does not do it justice

New Teacher Oriented Moodle MOOC

- Will be free to use
- Cover all Theory and Programming
- Include repository of shared resources
- Built in teacher assessment
  - When downloaded for use locally
- CPD Courses/Videos etc.
  - Best Practice
  - New CPD Courses now being made available through our CPD Hub

How to keep in touch

- OCR Website – www.ocr.org.uk
- Twitter: @ocr_ict
  - tweet/follow for resources/ideas/news/updates etc
- We are on CAS
- Facebook A Level Group: https://www.facebook.com/groups/625239200911674/
- **Customer Contact Centre**
  - Tel: 01223 553 998
  - Email: computerscience@ocr.org.uk

Please talk to us! We love to hear about:

- Ideas for resources
- Good sites for sharing
- Comments on improving our service etc!
- Positive feedback!