A2 Computer Science NEA Checklist

**Analysis**

1. Describe problem in a clear statement
2. Explanation of research you’ve done (interview/observation/internet searches)
3. End user details
4. Detailed explanation of current system and the problem you are solving
5. Objectives
6. Potential and Proposed solutions
7. E-R diagrams
8. DFDs

Remember:

* Researched the problem thoroughly
* Has clearly defined the problem being solved/investigated
* Statements of objectives which clearly and unambiguously identify the scope of the project.
* Modelled the problem for the Design stage where this is possible and necessary.

**Design**

1. Clear explanation of what your new solution will do (remember the examiner will appreciate as much detail as possible)
2. Clear and annotated user interface designs (Where design of a user interface is relevant, screen shots of actual screens are acceptable.
3. Choose 6-7 algorithms in your code and design them in pseudocode/flowcharts explaining how they fit into your solution.
4. Any extra information/diagrams (SQL notations, data structures, any validation)
5. Can you create a prototype and get some feedback from your end user?
6. Design a test plan
   1. should take the form of carefully selected and representative samples
      1. an introduction and overview
      2. the test performed
      3. its purpose if not self-evident
      4. the test data
      5. the expected test outcome
      6. the actual outcome with a sample of the evidence, for example screen shots of before and after the test, etc. sampled in order to limit volume.
   2. must include normal, erroneous and extreme tests
   3. also link your tests back to your objectives

**Technical Solution**

1. Copy your code (ensure it has comments)
2. Your code must be annotated and can take the form of any of these:
   1. an overview guide which amongst other things includes the names of entities such as executables, data filenames/urls, database names, pathnames so that a third party can, if they so desire, run the solution/investigation
   2. explanations of particularly difficult-to-understand code sections; a careful division of the presentation of the code listing into appropriately labelled sections to make navigation as easy as possible for a third party reading the code listing.

**Testing**

1. Use the test plan from the design section and complete the tests showing corrections where necessary (must include screenshots of them working/not working)
   1. demonstrate the **robustness** of the complete, or nearly complete, solution/**thoroughness** of investigation and which demonstrate that the requirements of the solution/investigation have been **achieved**.
2. Explain the tests that were carried out alongside the evidence for them (so when you had errors in your code and how you fixed them)
3. Possibly hand back to end user and get their feedback

**Evaluation**

1. Copy and paste the objectives and say whether they were successful or not
2. You need to obtain independent feedback on how well the outcome meets its requirements and discuss this feedback.
   1. Some of this feedback could be generated during prototyping. If so, this feedback, and how/ why it was taken account must be presented and referenced so it can be found easily.
3. You should also consider and discuss how your solution could be improved more realistically if the problem/investigation were to be revisited or extended.