P1 - Explain how computational thinking skills are applied in finding solutions that can be interpreted into software applications.

Explain what computational thinking is and analyse how programmers use it to solve problems, including how programmers identify and describe problems and how they communicate features and processes. How programmers recognise patterns, generalise and abstract information required, and how programmers represent problems or systems. **From the assignment brief.**

What is computational thinking?

What is a software application – give some examples of categories and maybe one or two examples from each category.

Category 1 – Admin – Office Suits and Email Applications (Libre Office Outlook Mail).

Category 2 – Web Browsers – Google Chrome and Firefox.

What are the computational thinking skills? – **Be sure to reference each one.**

Decomposition – What is it and how it useful in problem solving?

Patter recognition – What is it and how it useful in problem solving?

Pattern generalisation – What is it and how it useful in problem solving?

Partial Representation – What is it and how it useful in problem solving?

How are the above skills used to turn produce software applications?

How is Decomposition applied?

How is Patter recognition applied?

How is Pattern generalisation applied?

How is Partial Representation applied?

P2 - Explain how principles of computer programming are applied in different languages to produce software applications.

Explore the features and characteristics of programming languages, to explain the use of different types of programming language, what particular problems each programming language discussed can be used to solve, giving a comparison of those programming languages.

What is computer programming?

What are the principles of computer programming?

- Procedural – What is it and the top languages that use it.
- Object oriented – What is it and the top languages that use it.
- Event driven – What is it and the top languages that use it.
- Machine – What is it and the top languages that use it.
- Markup – What is it and the top languages that use it.

P3 - Explain how the principles of software design are used to produce high-quality software applications that meet the needs of users.

Describe the constructs and techniques available in different programming languages, explain how they are implemented and documented, contrasting their implementation in different programming languages.

What is software design?

What are the principles of software design? How is each principles used?

- Iteration
- Mathematical Logic
- Propositional Dynamic Logic
- Use of Sets

Gaming and Entertainment – brief description with a few examples, how to they solve the solve the needs of the person?

Productivity – brief description with a few examples.

Information Storage and management – brief description with a few examples.

Repetitive tasks or dangerous tasks – brief description with a few examples.

Social Media – brief description with a few examples.

Search engines – brief description with a few examples.

M1 - Analyse how computational thinking skills can impact software design and the quality of the software applications produced.

D1 - Evaluate how computational thinking skills can impact software design and the quality of the software applications produced.

References

What is Computational Thinking
Date Assessed: 29.09.2022
Link to BBC Bitesize: https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1

What is Computational Thinking

Date Accessed: 29.09.2022

Link to Information

Unit 4 Programming – Assignment B and C

From The Specification Part 1

Learning aim B: Design a software solution to meet client requirements

B1 Software development life cycle

Application of the software development life-cycle stages, including:

- assessment of the requirements for an identified problem
- design specification, e.g. scope, inputs/outputs, user interface, timescales
- develop code
- implementation
- test, e.g. white box and black box testing, refinement, optimisation
- maintenance, e.g. corrective, adaptive and increased functionality.

B2 Software solutions design

- Problem definition statements, to include: intended users, full summary of the problem to be solved, constraints, benefits, nature of interactivity, complexity of problem.
- Purpose and any other requirements as defined in a client brief.
- Features of software:
  - description of main program tasks, input and output formats
  - diagrammatic illustrations, to include screen layouts, user interfaces, navigation
  - algorithms and processing stages, to include flowcharts, pseudocode and events
  - data structures
  - data storage
  - control structures
  - data validation
  - error handling and reporting.
- Choice of language.
- List of pre-defined programs and/or code snippets.
- List of ready-made and/or original assets such as a digital animation, digital graphic, digital audio and video.
- Feedback from others to help refine alternative design ideas/prototypes and make decisions.
- Test plan with test data to include typical, extreme and erroneous data.
- Technical and design constraints, e.g. connectivity, memory storage, programming languages.

From The Specification Part 2

Learning aim C: Develop a software solution to meet client requirements

C1 Software solutions development

The process of software development, including:

- the development environment to produce code
- the development and refinement of software programs using a suitable programming language
- library routines, standard code and user generated subroutines used to add to the efficiency of a program.

C2 Testing software solutions

Testing of the programs, including:

- test plan
- test data – typical, extreme and erroneous data
- selection and use of appropriate types of testing to test part or all of a program, e.g. functional testing, stability, compatibility.

From The Specification Part 3

C3 Improvement, refinement and optimisation of software applications

Methods of improving, refining and optimising, e.g.:

- annotated code to allow effective repair/debugging of the program and maintainability
- program compilation for a designated platform or environment
- review – quality of a program in terms of reliability, usability, efficiency/performance, maintainability, portability
- eliciting feedback from users
- making use of the outcomes of testing and feedback
- documenting changes to the design and solution.

C4 Review of software solutions

Evaluation of software solutions, including:

- suitability for audience and purpose
- ease of use
- quality of the software solution, e.g. reliability, usability, efficiency/performance, maintainability, portability
- constraints of the programming language
- other constraints, e.g. time, programmer knowledge, rules of languages vary with implementation
- strengths and weaknesses of the software solutions
- improvements that can be made
- optimising software solutions, e.g. improving robustness, improving efficiency of the code, adding additional functionality.

C5 Skills, knowledge and behaviours

- Planning and recording, including the setting of relevant targets with timescales, how and when feedback from others will be gathered.
- Reviewing and responding to outcomes, including the use of feedback from others, e.g. IT professionals and users who can provide feedback on the quality of the program and its suitability when assessed against the original requirements.
- Demonstrating own behaviours and their impact on outcomes, to include professionalism, etiquette, supportive of others, timely and appropriate leadership, accountability and individual responsibility.
- Evaluating outcomes to help inform high-quality justified recommendations and decisions.
- Evaluating targets to obtain insights into own performance.
- Media and communication skills, including:
  - the ability to convey intended meaning, e.g. written (email, design documentation, recording documentation, reports, visual aids for presentation use); verbal communication requirements (one to one and group informal and formal situations)
  - use of tone and language for verbal and written communications to convey intended meaning and make a positive and constructive impact on audience, e.g. positive and engaging tone, technical/ vocational language suitable for intended audience, and avoidance of jargon
  - responding constructively to the contributions of others, e.g. supportive, managing contributions so all have the opportunity to contribute, responding to objections, managing expectations and resolving conflict.

From Brief

Produce a design for the tournament scoring system application including clear and effective diagrams, illustrations and algorithm designs. You will produce a design report in which you will:

- discuss software development life cycle stages, considering what areas of design and development should happen in which stages. You will produce an assessment of the scoring systems requirements and a design specification before any code is developed
- document the design of the system you will create, including descriptions the tasks your program needs to fulfil, algorithms your program will use, data structures and data storage needed by the system
- you should ensure that all of your diagrams and illustrations are relevant and accurately describe the programs you intend to create
- analyse the design options for the system, considering the features of the software you will create
- you should consider the advantages and drawbacks of using certain programming languages, identify any pre-defined code and assets available for use and how it could be integrated into the new system
- review your designs with others to obtain feedback and identify areas for improvement to evaluate and justify your final design
- using appropriate methods, compile a test plan with test data for the system to be tested against once development is complete.

Following the design you will develop the tournament scoring system application. You will implement the program to provide the functionality required by the college. You will produce a development report in which you will:

- demonstrate your use of a development environment and the chosen programming language, including the use of any pre-defined code and library routines within your program identifying how they improve program efficiency
- run your test plans from the design stage, ensuring that the program is thoroughly tested and that any errors found are documented with reasons why the error occurred and suggestions for repair
- repair errors found during the testing process with clear documentation for how repairs were made and results of retesting
- document errors that cannot be repaired, giving reasons why this is the case and suggest repairs for future reference
- review your program following feedback from users to identify areas for improvement and optimisation and prioritise which improvements to make with regard the time frame available to you.
- evaluate your final product covering how the decisions from all stages of the design and development process have ensured that the computer program produced, in comparison to other possible solutions, resulted in solutions that fully meet the college's requirements and the impact these processes had on the effectiveness of the development of the final outcomes.

You also need to show how you have taken individual responsibility and effectively managed yourself while completing this assignment. For example, you need to show how you have:

- planned and managed your time and met targets.
- reviewed and responded to outcomes including the use of feedback from others
- behaved appropriately while completing the assignment – including professionalism, etiquette, supportive of others, timely and appropriate leadership, accountability and individual responsibility
- evaluated outcomes to help inform high-quality justified recommendations and decisions
- used appropriate methods of communication effectively

Evidence Checklist

- all of your design documents such as, diagrams, pseudo-code and illustrations.
- records of review discussions (what was discussed and what decisions were made?)
- test plans (what will be tested and how?)
- program code
- program files (your working program)
- test logs (results of your test)
- error reports (what went wrong and how it was fixed)
- optimisation logs (what was improved)
- your evaluation of the development and the completed program
- a document which demonstrates that you have shown individual responsibility and effective self-management

Criteria

4/BC.D3	Demonstrate individual responsibility, creativity and effective self-management in the design, development and review of the computer program
4/BC.D2	Evaluate the final design and optimised software application against client requirements
4/B.M2	Justify design decisions, showing how the design will result in an effective solution
4/C.M3	Optimise the computer program to meet client requirements
4/B.P4	Produce a design for a computer program to meet client requirements
4/B.P5	Review the design with others to identify and inform improvements to the proposed solution
4/C.P6	Produce a computer program that meets client requirements
4/C.P7	Review the extent to which the final computer program meets client requirements

Pre P4 Stuff (Requirements, Specification and Planning)

Requirements and specification section AND the planning section.

Define everything.

What are requirements? With reference.

What is a specification? With reference.

After doing that you outline you requirements and specification.

Requirements can almost be copied from the assignment brief's scenario.

This is what needs to be done by you.

Requirements are simply an interpretation of what needs to be done.

Google Search: How to write a specification?

What is project management? – REF it.

Why is project management important? – REF it.

What are some project management methods? – REF it.

Which do you think will work best for your project? Give some information on this.

Why do you think that will work well compared to others, the benefits?

Requirements and Specifications

- **Participants may enter the tournament as individuals or as part of a team.**
- Have the option available to allow a single person to be a team or have multiple people be a team. When the program runs ask if a team or single person.

- **It is expected that will be 4 teams each with 5 members and there will be 20 spaces for individual competitors.**
- Allow each team to enter five names, these will be the names of the team members.

- **Each team or individual will complete 5 events.**
- Each team whether it be a single person or five people will play five games in total. This can be a manual process but an automatic selection and playing will be explored.

- **Each event will be defined as a team or individual event.**
- Every event that is played will be labelled a team event or an individual event.

- **The events will vary in type, from sporting to academic challenges.**
- Have numerous games in the event, possibly have the judge choose the games at the start or the five games can be chosen at random. Another option is to have a list of possible games, have the program choose at random five different games and those are the options.

- **Individuals and teams will be awarded points according to their rank within each event.**
- **The points awarded for each event are as yet undecided and the college are willing to hear any suggestions you may have.**
- Have a scoring system in place which will give appropriate scores to the participants. Since there are five teams. First place gets 5 points. Second place gets 4 points. Third place 3 points. Second place 2 points and last place 1 point or zero points.

- **Also the college would like to include the possibility of entering for one event only.**
- Have it possible for a team to choose only one event.

You now have a nice check list of things that you need to complete.
You can refer to this when it comes to evaluating what was done.
For example, 7 things were required, you might have only done 4 of the 7.

Project Planning

What is project planning and why is it important?

Your project plan will come next.

Initiation

Plan

Design

Implement

Test

Evaluation

Plan for Design

What is design and why is it important?

I plan to use pseudocode and flowcharts.

What is pseudocode?

What is a flowchart.

What will be done and why?

For the pseudocode I will simply use a word processor, like Microsoft word or google Docs. Why?

For the Flowcharts I will use PowerPoint. Why? It has all the shapes and connectors I need.

Pseudocode will simply be copied into my document.

Flowchart will be exported as a PNG or screenshotted and imported into the document.

If you want to be fancy, you can sketch a GUI as well.

Design pieces of evidence: pseudocode text, flowchart diagram, images of GUI sketch.

Plan for Implementation/Development

What is project implementation and why is it important?

What and why just like before.

Python and why? – any other is fine (Java, C, C++, Visual Basic).

Chrome and why? – Python online, python cloud, no software to install, work anywhere, low spec computers.

Windows and why?

Plan for Testing and Evaluation

What is testing and why is it important?

Types of testing.

Which testing method will you use?

What is black box testing?

What is white box testing?

I would use both white and black box testing. Say why.

I will test if a feature works I will evidence this (screenshot) and tick that of from requirements.

If it does not work I will evidence this (screenshot) and try to rectify the issue.

If the issue is rectified I will evidence how it was fixed.

If not fixed I will remove that feature in order to prevent crashing.

I will also ask others to review my designs and development. Why? This might aid me in fault finding and general improvements.

P4 - Produce a design for a computer program to meet client requirements.

Review requirements and specification.

You can only design if you know what you are designing for.

A bullet pointed list is fine.

A paragraph is fine.

	Program starts and displays a welcome message to the user.
	There is a short pause, and all the possible games are displayed to the user.
	Team 1 is asked if a single person or a team. Enter name/s and press enter.
	Team 2 is asked if a single person or a team. Enter name/s and press enter.
	Team 3 is asked if a single person or a team. Enter name/s and press enter.
	Team 4 is asked if a single person or a team. Enter name/s and press enter.
	Team 5 is asked if a single person or a team. Enter name/s and press enter.
	There is a pause and teams and all members are listed.
	The user is asked to type “some value” to continue.
	Game 1 is chosen at random from the list and that game is removed.
	All teams play the game and scores are allocated.
	Game 2 is chosen at random from the list and that game is removed.
	All teams play the game and scores are allocated.
	Game 3 is chosen at random from the list and that game is removed.
	All teams play the game and scores are allocated.
	Game 4 is chosen at random from the list and that game is removed.
	All teams play the game and scores are allocated.
	Game 5 is chosen at random from the list and that game is removed.
	All teams play the game and scores are allocated.
	The scores of all the teams are shown at the end, winning team is congratulated.
	Not perfect but shows some of the actions that need to be completed. Add more if you need to.

Pseudocode

Keywords

- **INPUT** – indicates a user will be inputting something
- **OUTPUT** – indicates that an output will appear on the screen
- **WHILE** – a **loop** (**iteration** that has a **condition** at the beginning)
- **FOR** – a counting loop (iteration)
- **REPEAT – UNTIL** – a loop (iteration) that has a condition at the end
- **IF – THEN – ELSE** – a decision (**selection**) in which a choice is made
- any instructions that occur inside a selection or iteration are usually indented

BBC Bitesize: Pseudocode.

RonsTechHub

Flowchart

Common flowchart symbols

Name	Symbol	Usage
Start or Stop		The beginning and end points in the sequence.
Process		An instruction or a command.
Decision		A decision, either yes or no.
Input or Output		An input is data received by a computer. An output is a signal or data sent from a computer.
Connector		A jump from one point in the sequence to another.
Direction of flow		Connects the symbols. The arrow shows the direction of flow of instructions.

[BBC Bitesize: Flowcharts.]

Open PowerPoint and design flowchart.

P5 - Review the design with others to identify and inform improvements to the proposed solution.

What is a review and why is it important?

Why are you reviewing?

Pseudocode Review

Person 1 – Bob The Builder

Good Points	Needs Improving

Person 2 – Dora The Explorer

Good Points	Needs Improving

Person 3 – Son Goku

Good Points	Needs Improving

Person 4 – Ichigo Kurosaki

Good Points	Needs Improving

Flowchart Review

Person 1 – Bob The Builder

Good Points	Needs Improving

Person 2 – Dora The Explorer

Good Points	Needs Improving

Person 3 – Son Goku

Good Points	Needs Improving

Person 4 – Ichigo Kurosaki

Good Points	Needs Improving

Sketches Review

Person 1 – Bob The Builder

Good Points	Needs Improving

Person 2 – Dora The Explorer

Good Points	Needs Improving

Person 3 – Son Goku

Good Points	Needs Improving

Person 4 – Ichigo Kurosaki

Good Points	Needs Improving

Changes To Be Made (Pseudocode)

More detail, show all the steps in a simple way for the repeats.

More detail, show all the steps in a simple way for the repeats.

More detail, show all the steps in a simple way for the repeats.

More detail, show all the steps in a simple way for the repeats.

Changes To Be Made (Flowchart)

Update to show all the changes made in Pseudocode.

Changes To Be Made (Sketches)

Updated Pseudocode After Review

Updated Flowchart After Review

Updated Sketches

M2 - Justify design decisions, showing how the design will result in an effective solution.

Problem definition statements, to include: intended users, full summary of the problem to be solved, constraints, benefits, nature of interactivity, complexity of problem.

Interactivity

What is this? Why design in this way?

Answer: GUI or Command line, simple number choices, simple, easy to follow

Variables what are they and why did you use them?

How will using this result in an effective solution?

Data Types what are they and why did you use them?

How will using this result in an effective solution?

Sequence what are they and why did you use them?

How will using this result in an effective solution?

Selection what are they and why did you use them?

How will using this result in an effective solution?

Iteration what are they and why did you use them?

How will using this result in an effective solution?

Data Structures what are they and why did you use them?

How will using this result in an effective solution?

Data Storage what are they and why did you use them?

How will using this result in an effective solution?

Control Structures what are they and why did you use them?

How will using this result in an effective solution?

Data Validation what are they and why did you use them?

How will using this result in an effective solution?

Error Handling and Reporting what are they and why did you use them?

How will using this result in an effective solution?

- Choice of language.
- List of pre-defined programs and/or code snippets.
- List of ready-made and/or original assets such as a digital animation, digital graphic, digital audio and video.
- Feedback from others to help refine alternative design ideas/prototypes and make decisions.
- Test plan with test data to include typical, extreme and erroneous data.
- Technical and design constraints, e.g. connectivity, memory storage, programming languages.

Test Plan what are they and why did you use them?

How will using this result in an effective solution?

P6 - Produce a computer program that meets client requirements.

file:///C:/Users/ronal/Desktop/IT%20-%20Unit%204/Unit%204%20Spec.pdf

Learning aim C: Develop a software solution to meet client requirements

C1 Software solutions development

The process of software development, including:

- the development environment to produce code
- the development and refinement of software programs using a suitable programming language
- library routines, standard code and user generated subroutines used to add to the efficiency of a program.

P6 + M3 at the same time??

Python time, lets go.

Use your pseudocode and/or flowchart to help you.

Refer to the plan.

Are you going to use what you said you would?

I am using PyCharm instead of Thonny, for example.

Do one section at a time.

Outputs to introduce game.

Get user input.

Play game and record scores.

Etc.

I would then copy the entire code into your document OR a section at a time.

Explain each major section with either comments or a text under the screen shots.

PASTE CODE HERE – do not make it too perfect, you will need to review and optimise.

Non efficient.

Non commented.

No functions.

M3 - Optimise the computer program to meet client requirements.

file:///C:/Users/ronal/Desktop/IT%20-%20Unit%204/Unit%204%20Spec.pdf

C3 Improvement, refinement and optimisation of software applications

Methods of improving, refining and optimising, e.g.:

- annotated code to allow effective repair/debugging of the program and maintainability
- program compilation for a designated platform or environment
- review – quality of a program in terms of reliability, usability, efficiency/performance, maintainability, portability
- eliciting feedback from users
- making use of the outcomes of testing and feedback
- documenting changes to the design and solution.

What does optimise mean?

How do you optimise a computer program? What is meant by code/program optimisation?

Google Search: How to Optimise a computer program.

Screenshot from the spec.

How did you optimise?

User feedback

Own review and own feedback

Results from testing

Comments code- Python is “#”

Making some things a little better

Annotated code simply means add comments.

Screenshot before the optimise AND after the optimise.

Explain what was done and why.

BEFORE

```
# all the scores of all the teams
team1scores = 0
team2scores = 0
team3scores = 0
team4scores = 0
team5scores = 0
```

AFTER

```
# single lsit for alls cores
all_team_scores = [10,30,10,50,90]
```

WHY?

BEFORE

```
print("Team 1 Check Value:",team1check,)
print("Team 1 Type Value:",team1type)
print("Team 1 Names Value:",team1names)
print("Number Of Games:",len(allgames))
print(allgames)
```


AFTER

```
print("Team 1 Check Value:",team1check, "Team 1 Type Value:",team1type,"Team 1 Names Value:",tea
```

WHY?

WINDOWS + SHIFT + S

RonsTechHub

P7 - Review the extent to which the final computer program meets client requirements.

D2 - Evaluate the final design and optimised software application against client requirements.

C4 Review of software solutions

Evaluation of software solutions, including:

- suitability for audience and purpose
- ease of use
- quality of the software solution, e.g. reliability, usability, efficiency/performance, maintainability, portability
- constraints of the programming language
- other constraints, e.g. time, programmer knowledge, rules of languages vary with implementation
- strengths and weaknesses of the software solutions
- improvements that can be made
- optimising software solutions, e.g. improving robustness, improving efficiency of the code, adding additional functionality.

Copy the requirements here along with the specification.

I would do a bullet pointed list.

Task Name/Description	Completeness
Game welcome message, describing the games etc.	Completed.
Get user input from teams.	Completed.
Single person allowed?	Not Completed.
Teams allowed?	Not Completed.

Give screenshots of the program output at each stage.

Welcome message.

User input team or person.

First game played

```
Welcome To The Most Amazing quiz Game Ever...
Teams can be a single person or a team of five people.

The games that can be played are: ['Math', 'History', 'Computer Science', 'IT', 'Business', 'Engineering
']
```

Hjhbfvkvjhfbfjfhbvdkfvdjfjhbvjhf kjhfbjh ffg uudfgu urhurudrdu drudgrukrdr

Suitable for Audience

Yes.

Though not completed the program allows users to play the games and keep track of the scores using an automated system.

Ease Of Use

Anyone with a basic understanding of the English language should be able to play the game. The system uses a simple Command Line Interface (CLI) to ask the user some questions.

Users are then given the choice to use simple input options. For example, if they wish to play as a team type the letter “t” and the letter “p” if they wish to play as a person.

Instructions are simple and are spread out both in space and time allowing for the users to easily follow.

WIMP

Quality Of Software Solution

Google Search: What makes quality software?

Programming Language Constraints and Other

Strengths and Weaknesses Of Software Solution

WIMP

Possible Improvements

RonsTechHub

D2 - Evaluate the final design and optimised software application against client requirements.

file:///C:/Users/ronal/Desktop/IT%20-%20Unit%204/Unit%204%20Spec.pdf

C4 Review of software solutions

Evaluation of software solutions, including:

- suitability for audience and purpose
- ease of use
- quality of the software solution, e.g. reliability, usability, efficiency/performance, maintainability, portability
- constraints of the programming language
- other constraints, e.g. time, programmer knowledge, rules of languages vary with implementation
- strengths and weaknesses of the software solutions
- improvements that can be made
- optimising software solutions, e.g. improving robustness, improving efficiency of the code, adding additional functionality.

Just like we did before.

Ask people to look at it.

I would show them the running program, not the code, to lower the possibility of cheating.

You have the cleint's requirements.

Make a list of them, make the list micro requirements (break them down as much as you can),

Then simply tick things off that are actually being done by the program.

Requirements	Yes or No
Welcome message	
Single person allowed	
Teams allowed	
Single person name taken	
Team names taken	

D2 - Evaluate the final design and optimised software application against client requirements.

D3 - Demonstrate individual responsibility, creativity and effective self-management in the design, development and review of the computer program.

C5 Skills, knowledge and behaviours

- Planning and recording, including the setting of relevant targets with timescales, how and when feedback from others will be gathered.
- Reviewing and responding to outcomes, including the use of feedback from others, e.g. IT professionals and users who can provide feedback on the quality of the program and its suitability when assessed against the original requirements.
- Demonstrating own behaviours and their impact on outcomes, to include professionalism, etiquette, supportive of others, timely and appropriate leadership, accountability and individual responsibility.
- Evaluating outcomes to help inform high-quality justified recommendations and decisions.
- Evaluating targets to obtain insights into own performance.
- Media and communication skills, including:
  - the ability to convey intended meaning, e.g. written (email, design documentation, recording documentation, reports, visual aids for presentation use); verbal communication requirements (one to one and group informal and formal situations)
  - use of tone and language for verbal and written communications to convey intended meaning and make a positive and constructive impact on audience, e.g. positive and engaging tone, technical/ vocational language suitable for intended audience, and avoidance of jargon
  - responding constructively to the contributions of others, e.g. supportive, managing contributions so all have the opportunity to contribute, responding to objections, managing expectations and resolving conflict.

file:///C:/Users/ronal/Desktop/IT%20-%20Unit%204/Unit%204%20Spec.pdf

print("Hello World")