Government of **Western Australia**
School Curriculum and Standards Authority

Sample Teaching and Learning Outline

Technologies

Digital Technologies

Year 10

Disclaimer

Any resources such as texts, websites and so on that may be referred to in this document are provided as examples of resources that teachers can use to support their teaching and learning programs. Their inclusion does not imply that they are mandatory or that they are the only resources relevant to the learning area syllabus.

2020/18989v4

This document is an introduction to planning a teaching and learning outline with syllabus content for Year 10 Digital Technologies. It provides suggested sequencing and timing for teaching the syllabus content. For further details on curriculum requirements and available options, teachers should refer to the School Curriculum and Standards Authority's (the Authority's):

- *Policy Standards for Pre-primary to Year 10: Teaching, Assessing and Reporting*
- Table 1: *Western Australian Curriculum and Assessment Outline*: curriculum requirements and available options.

Schools may choose to teach the syllabus content for two hours a week for a semester **or** one hour per week for the year. Sample plans provide a range of possible learning experiences from which assessment should be drawn. This *Year 10 Sample Teaching and Learning Outline* provides teachers with possible learning experiences over nineteen weeks and unpacks the syllabus content to assist teachers in their understanding.

A presentation (*Western Australian Curriculum Technologies Presentation*), which unpacks the process to develop this plan, is available on the Presentations page of the Authority website (https://k10outline.scsa.wa.edu.au/home/resources/presentations).

Year 10 Syllabus Content – Digital Technologies

Content	Description
Digital systems	Role of hardware and software in managing, controlling and securing access to data, in networked digital systems
Representation of data	Simple compression of data and how content data is separated from presentation data
Collecting, managing and analysing data	Apply techniques for acquiring, storing and validating quantitative and qualitative data from a range of sources, considering privacy and security requirements Analyse, visualise and model processes and entities, and their relationships, using structured data
Digital implementation	Design algorithms represented diagrammatically and in structured English, including iteration Validate algorithms and programs using common acceptable methods Implement data storage and organisation techniques within a programming environment Create interactive solutions for sharing ideas and information online, taking into account social contexts and legal responsibilities
Investigating and defining	Identify the needs of the client/stakeholder to determine the basis for a solution Create and critique briefs to solutions Investigate components/resources to develop increasingly sophisticated solutions, identifying and considering associated constraints

Content	Description
Designing	Design possible solutions, analysing designs against criteria, including functionality, accessibility, usability and aesthetics using appropriate technical terms and technology Apply design thinking, creativity, enterprise skills and innovation to develop, modify and communicate design ideas of increasing sophistication
Producing and implementing	Select, justify, and safely implement and test appropriate technologies and processes, to make solutions
Evaluating	Analyse design processes and solutions against student developed criteria
Collaborating and managing	Work independently, and collaboratively, to manage projects, using digital technology and an iterative and collaborative approach. Considers time, cost, risk, safety, production processes, sustainability and legal responsibilities

Year Level Description

In Year 10, learning in digital technologies focuses on further developing understanding and skills in computational thinking, such as precisely and accurately describing problems; and the use of modular approaches to solutions. It also focuses on engaging students with specialised learning in preparation for vocational training or learning in the senior secondary years.

Students have opportunities to analyse problems and design, implement and evaluate a range of solutions, such as database-driven websites, artificial intelligence engines and simulations.

Students consider how human interaction with networked systems introduces complexities surrounding access to, and the security and privacy of, data of various types. They interrogate security practices and techniques used to compress data, and learn about the importance of separating content, presentation and behavioural elements for data integrity and maintenance purposes.

Students explore how bias can impact the results and value of data collection methods, and use structured data to analyse, visualise, model and evaluate objects and events.

Students learn how to develop multilevel abstractions; identify standard elements, such as searching and sorting in algorithms; and explore the trade-offs between the simplicity of a model and the faithfulness of its representation.

When defining problems, students consider the functional and non-functional requirements of a solution through interacting with clients/stakeholders and regularly reviewing processes. They consolidate their algorithmic design skills to incorporate testing and review, and further develop their understanding of the user experience to incorporate a wider variety of user needs. Students develop solutions to complex problems and evaluate their solutions and existing information systems, based on a broad set of criteria, including connections to existing policies and their enterprise potential. They consider the privacy and security implications of how data are used and controlled, and suggest how policies and practices can be improved to ensure the sustainability and safety of information systems.

Students have opportunities to become more skilled at identifying the steps involved in planning solutions and developing detailed plans that are mindful of risks and sustainability requirements. When creating solutions individually, collaboratively and interactively for sharing in online environments, students should comply with legal obligations, particularly with respect to the ownership of information.

Year 10 Learning Area: Technologies – Digital Technologies

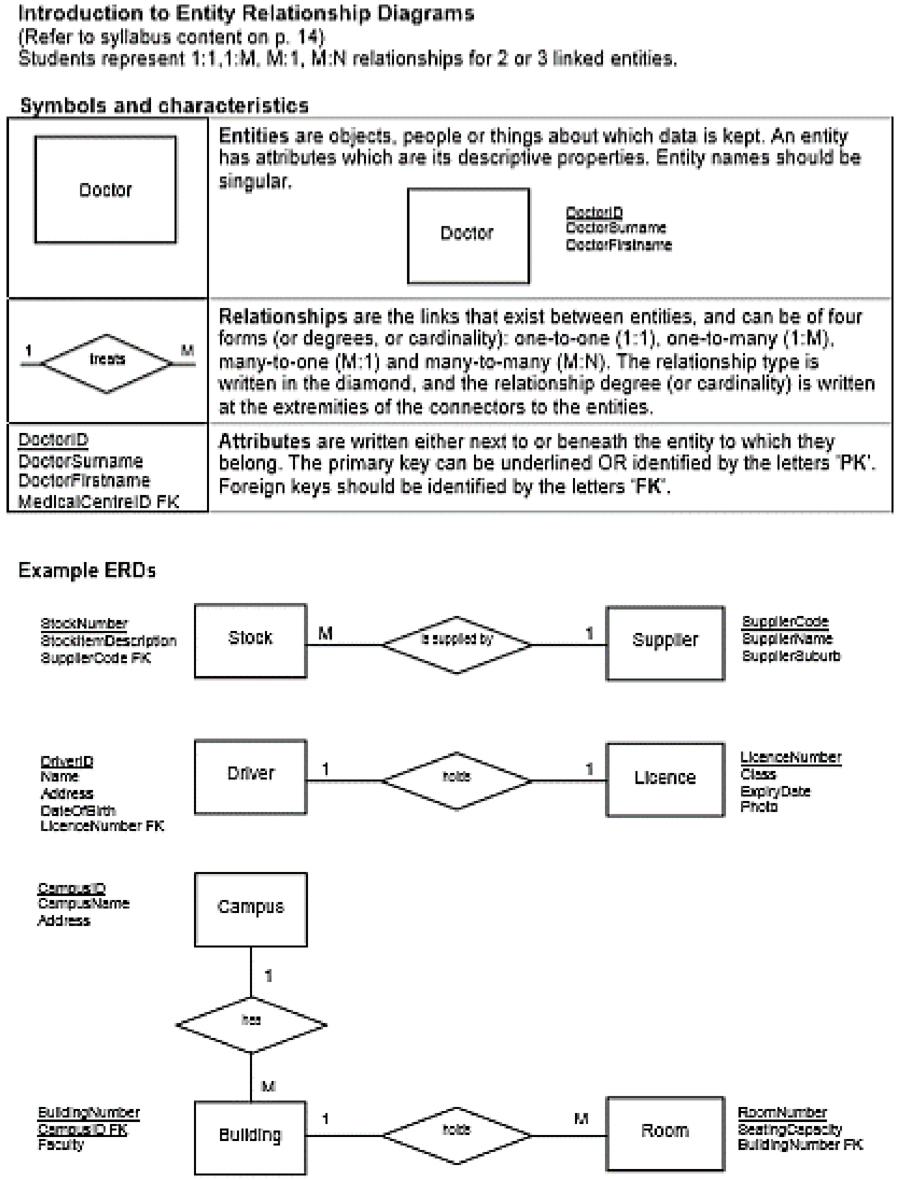
Year 10 Achievement Standard

At Standard, students describe the role of hardware and software in managing, controlling and securing access to data, in networked digital systems. They describe the process of simple compression of data and how content data is separated from presentation data. Students apply techniques for acquiring, storing and validating quantitative and qualitative data from a range of sources, and consider privacy and security requirements. They analyse, visualise and model processes and entities, and their relationships, using structured data. Students create a design for algorithms represented diagrammatically and in structured English, including iteration. They validate algorithms and programs, using commonly accepted methods. Students implement data storage and organisation techniques within a programming environment. They create interactive solutions for sharing ideas and information online, taking into account social contexts and legal responsibilities.

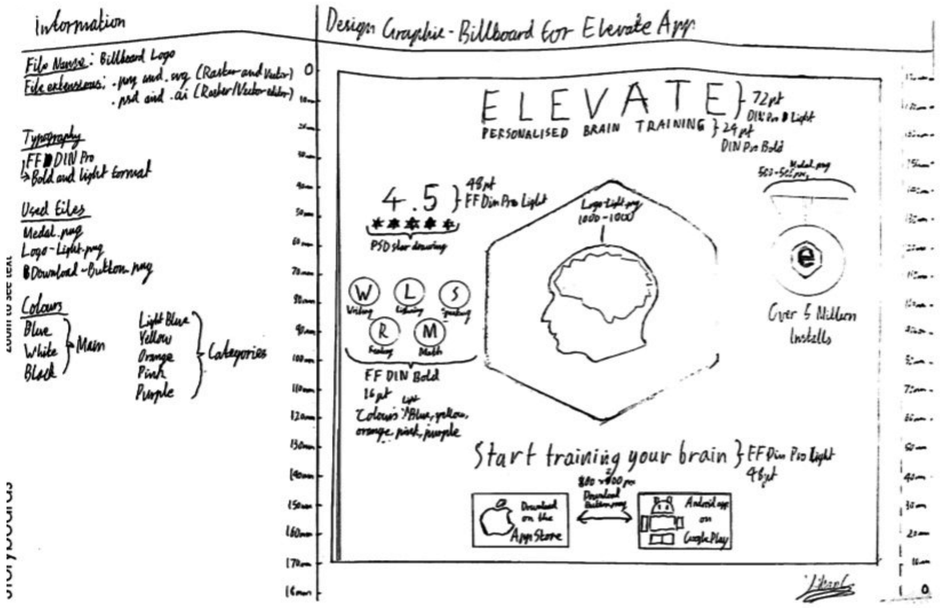
In digital technologies, students identify the needs of the client/stakeholder to determine the basis for a solution. They create and critique briefs. Students investigate components/resources to develop increasingly sophisticated solutions, identifying and considering associated constraints. They apply design thinking, creativity, enterprise skills and innovation to develop, modify and communicate design ideas of increasing sophistication. Students design possible solutions, analysing designs against criteria, including functionality, accessibility, usability and aesthetics, using appropriate technical terms and technology. They select, justify and safely implement and test appropriate technologies and processes to make solutions. Students provide relevant analysis of design processes and solutions against student-developed criteria. They work independently, and collaboratively to manage projects, using digital technology and an iterative and collaborative approach. Students consider time, cost, risk, safety, production processes, sustainability and legal responsibilities.

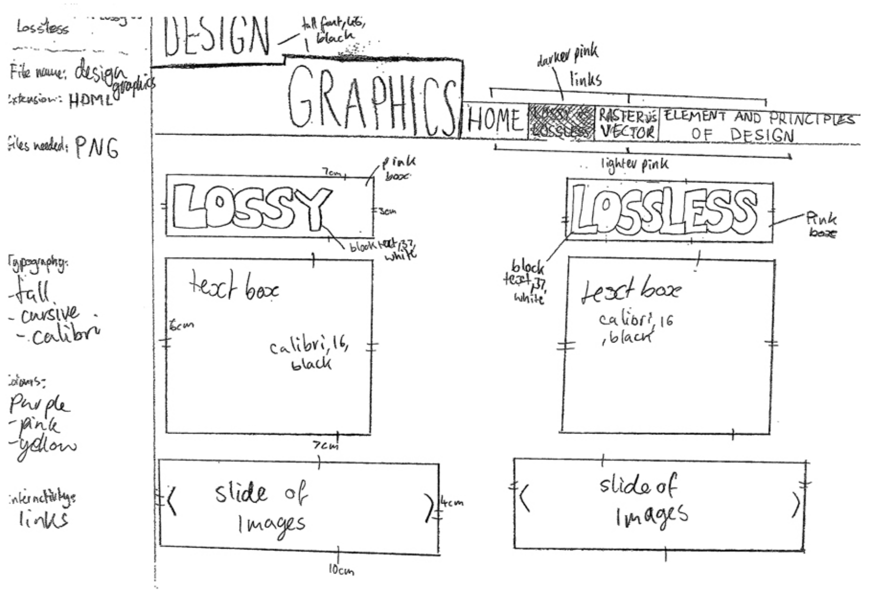
Weeks	Syllabus content	Content unpacked	Suggested teaching and learning experiences
1–3	**Digital systems** Role of hardware and software in managing, controlling and securing access to data, in networked digital systems **Collecting, managing and analysing data** Apply techniques for acquiring, storing and validating quantitative and qualitative data from a range of sources, considering privacy and security requirements	• Securing and controlling data can be carried out through a variety of physical and virtual tools. These can include: ▪ Encryption – the process of encoding and decoding a message so if it is intercepted in transmission it is difficult to read. ▪ Firewalls – a barrier to block unwanted traffic into a system through the identification of authorised and unauthorised IP addresses. ▪ MAC filtering – only allows data to be sent and received from network interface cards on devices in a network through their physical addresses. ▪ Password protection – passwords are used to access a network and to authenticate the user. Strong passwords include a combination of symbols, numbers, upper and lower-case letters. ▪ Physical security – biometrics, locks, security cameras etc. • Common security threats include: ▪ DDOS attacks – Distributed Denial Of Service attacks. Overwhelming a server by flooding it with internet traffic in order to disrupt its usual operations. ▪ Intrusion – unauthorised access to a system or network. ▪ Backdoor attacks – malware used to gain unauthorised access to a system through finding unsecured sections of a network. • *Privacy Act 1988* (Privacy Act): ▪ The law that governs the collection, use and disclosure of personal information. • Intellectual property is the law that protects intangible items such as ideas and designs. This allows for credit to be given to the creator and recognition of the originator of the idea. • Fair use is a term to describe the permission of the limited use of copyrighted material. This can only be imposed for purposes such as criticism, parody, journalism and court of law proceedings. • Quantitative: collection of facts and figures, e.g. 12/15. • Qualitative: information that is topic-based discovery, e.g. colour of cars in the car park. • Data sources include: observation, questionnaire, interview, sample forms.	Students explain the correlation between security threats and solutions. This should be broken into two lessons one for the threats and one for the solutions. Both lessons can be delivered with a combination of appropriate YouTube clips, definitions and examples. • Students research network security breaches to find a case study of their choice. Once a case study is selected, students explain the breach and the implications of the breach. A solution must be suggested and compared to the case study. • Students create a short animation explaining a threat or solution to network security. • Students research the 13 privacy principles and how they affect businesses. This can be completed in pairs or small groups with each group becoming an expert of one privacy principle. The groups then deliver the information to the rest of the class through a presentation, short video or an infographic. • Australian Government Office of the Australian Information Commissioner Australian Privacy Principles 13 Privacy Principles (https://www.oaic.gov.au/privacy/australian-privacy-principles/). • Students gather data from conducting observation, questionnaire, interview or sample forms. Data should relate to the *Privacy Act 1988*. Students can ask members of the community selected questions to gauge their understanding of the law. This information should be visualised, analysed and presented to the teacher in a creative way. • Students research and create an infographic on intellectual property law in Australia. This should include example of what it covers, its purpose, fair use and consequences of breaching the law https://www.ipaustralia.gov.au/about-us/legislation/ip-legislation. Useful links for lesson ideas: • ABC Education How is your information kept secret on the internet? (https://education.abc.net.au/home#!/media/2211736/how-is-your-personal-information-kept-secret-on-the-internet) • Digital Technologies Hub Network Security (https://www.digitaltechnologieshub.edu.au/resourcedetail?id=24b84698-09f9-6792-a599-ff0000f327dd#/) • Khan Academy Journey into Cryptography (https://www.khanacademy.org/computing/computer-science/cryptography) • Digital Technologies Hub Connected via a network (https://www.digitaltechnologieshub.edu.au/teachers/scope-and-sequence/9-10/digital-systems/connected-via-a-network).

Weeks	Syllabus content	Content unpacked	Suggested teaching and learning experiences
4	**Investigating and defining** Identify the needs of the client/stakeholder to determine the basis for a solution Create and critique briefs to solutions Investigate components/resources to develop increasingly sophisticated solutions, identifying and considering associated constraints **Collaborating and managing** Work independently, and collaboratively to manage projects, using digital technology and an iterative and collaborative approach. Considers time, cost, risk, safety, production processes, sustainability and legal responsibilities	• Project briefs can be created to identify the following elements of a project: ▪ scope ▪ intended purpose/client's needs ▪ hardware required ▪ software required ▪ time plan (Gantt chart) ▪ constraints ▪ cost ▪ risk ▪ legal responsibilities ▪ other as required.	Teacher lead instruction about the purpose and how to create time plans such as a Gantt chart. Activity should include the creation of a game and the stages of software development. This should focus on concurrent and sequential tasks. This will aid in an understanding of how long a game would take to design, produce and implement. Teacher could choose to include information about software updates and patches. Budgeting is an important tool for all software developers for task to be viable and cost effective. Based on the analysis completed in the above activity, students use excel formulas to create an estimated budget for the production. Teacher should provide estimations of costs such as labour. **E-sports database** • As a class, students explore concepts related to the collection of data and techniques used for data storage, including privacy and security considerations. Students also develop knowledge and skills associated with the development of a database using appropriate software. This task will be conducted in three parts. • In Part A, students are required to: ▪ conduct a survey to find out what member information is commonly collected by sporting or interest groups. Include the types of hardware and software required to gather and implement this information and data ▪ choose a sporting or interest group and design a membership database for that group ▪ complete a client brief based on the primary data collected. Useful links: • BBC Bitesize What is a database? (https://www.bbc.co.uk/bitesize/topics/zf2f9j6/articles/z8yk87h).
5–8	**Collecting, managing and analysing data** Analyse, visualise and model processes and entities, and their relationships, using structured data **Digital Implementation** Implement data storage and organisation techniques within a programming environment	• Entity relationship diagrams (ERDs) can be used to visualise the data and interactions between entities. • Structured Query Language (SQL), is a programming language used to manage, operate and query data from databases. These queries can input and output selected data with constraints.	**Database implementation skills** • Teacher develops student's skills in database development. Teachers and the school can choose software applications to use to create databases. This will aid students understanding and allow for database design to be more efficient. Database skills should include table creation, relationships, queries, reports or forms. **Entity relationship diagrams (ERD)** • Teacher revises the theory of ERDs, including purpose, characteristics and symbols • Class completes some examples as a class • Students work through examples and get feedback on their progress. **Structured Query Language (SQL)** • Teacher uses SQLite, or similar software, with a premade database to allow students to create queries on paper and to test these theories using the sample database provided. They should get progressively harder and include inputting information, creating joins, use of syntax, such as select, update, delete, order by, count and the use of like where appropriate.

Weeks	Syllabus content	Content unpacked	Suggested teaching and learning experiences
		 **Introduction to Entity Relationship Diagrams** (Refer to syllabus content on p. 14) Students represent 1:1,1:M, M:1, M:N relationships for 2 or 3 linked entities. **Symbols and characteristics** **Entities** are objects, people or things about which data is kept. An entity has attributes which are its descriptive properties. Entity names should be singular. **Relationships** are the links that exist between entities, and can be of four forms (or degrees, or cardinality): one-to-one (1:1), one-to-many (1:M), many-to-one (M:1) and many-to-many (M:N). The relationship type is written in the diamond, and the relationship degree (or cardinality) is written at the extremities of the connectors to the entities. **Attributes** are written either next to or beneath the entity to which they belong. The primary key can be underlined OR identified by the letters 'PK'. Foreign keys should be identified by the letters 'FK'. **Example ERDs** Doctor Doctor DoctorID DoctorSurname DoctorFirstname 1 treats M DoctorID DoctorSurname DoctorFirstname MedicalCentreID FK StockNumber StockItemDescription SupplierCode FK Stock M is supplied by 1 Supplier SupplierCode SupplierName SupplierSuburb DriverID Name Address DateOfBirth LicenceNumber FK Driver 1 holds 1 Licence LicenceNumber Class ExpiryDate Photo CampusID CampusName Address Campus 1 has M BuildingNumber CampusID FK Faculty Building 1 holds M Room RoomNumber SeatingCapacity BuildingNumber FK	**Esports database** • As a class, students explore concepts related to the collection of data and techniques used for data storage including privacy and security considerations. Students develop knowledge and skills associated with the development of a database using appropriate software. This task is conducted in three parts: Part A: students are required to: • conduct a survey to find out what member information is commonly collected by sporting clubs or interest groups. Include the types of hardware and software required to gather and implement this information and data • choose a sporting or interest group to design a membership database • complete a Client Brief based on primary data collected. In Part B: students create an accurate Entity Relationship Diagram (ERD): • design a database using an ERD for the team/sport of your choice. For E-sports, it should include player, team, game, tournament etc. In Part C: students create a relational database based on designs: • all tables should be linked. • all fields and records complete. • both the theory and practical SQL queries should be complete. • students may use SQLite to help learn the skills required to create basic queries.
9	**Designing** Design possible solutions, analysing designs against criteria, including functionality, accessibility, usability and aesthetics using appropriate technical terms and technology Apply design thinking, creativity, enterprise skills and innovation to develop, modify and communicate design	• Storyboards – graphical representation of the visual elements of a user interface. Used also in the animation and film design process. • Should include interactivity, functionality, navigation, useability elements, colours, sizes etc. • Functionality is the inclusion of a range of different operations and how well they perform, including the ability to attract and retain users. • Accessibility is making an interface as reachable to as many users as possible, including different skills, abilities, platforms and internet capabilities. • Useability is the ease in which an interface can be accessed or its effectiveness in achieving its intended purpose.	• Teacher selects a user interface where students can create a storyboard based on the current interface. This can be a popular website, the school website or an app of choice. Students redesign the interface to make it more functional, accessible and with increased useability. Useful links • The World's Worst Website Ever (https://www.theworldsworstwebsiteever.com/) **Programming – choose your own code-venture** • Students need to create an online subject selection payment system. This program needs to add up the cost of student courses in order to enrol them. • Mathematics, English, Science and Humanities and Social Sciences (HASS) subjects are compulsory (**constant**) but students can choose only two electives. Each elective has different values (**variables**). • Students should only be able to add two of the electives, and should use **data validation techniques** and **control structures** to do this. • Using the price list below, students must output student details as well as the total price for the next year. The design of the interface is up to the student, but must be storyboarded. • The task is broken up into five stages:

<table>
<tr><th>Weeks</th><th>Syllabus content</th><th>Content unpacked</th><th>Suggested teaching and learning experiences</th></tr>
<tr>
<td></td>
<td>ideas of increasing sophistication</td>
<td>





</td>
<td>

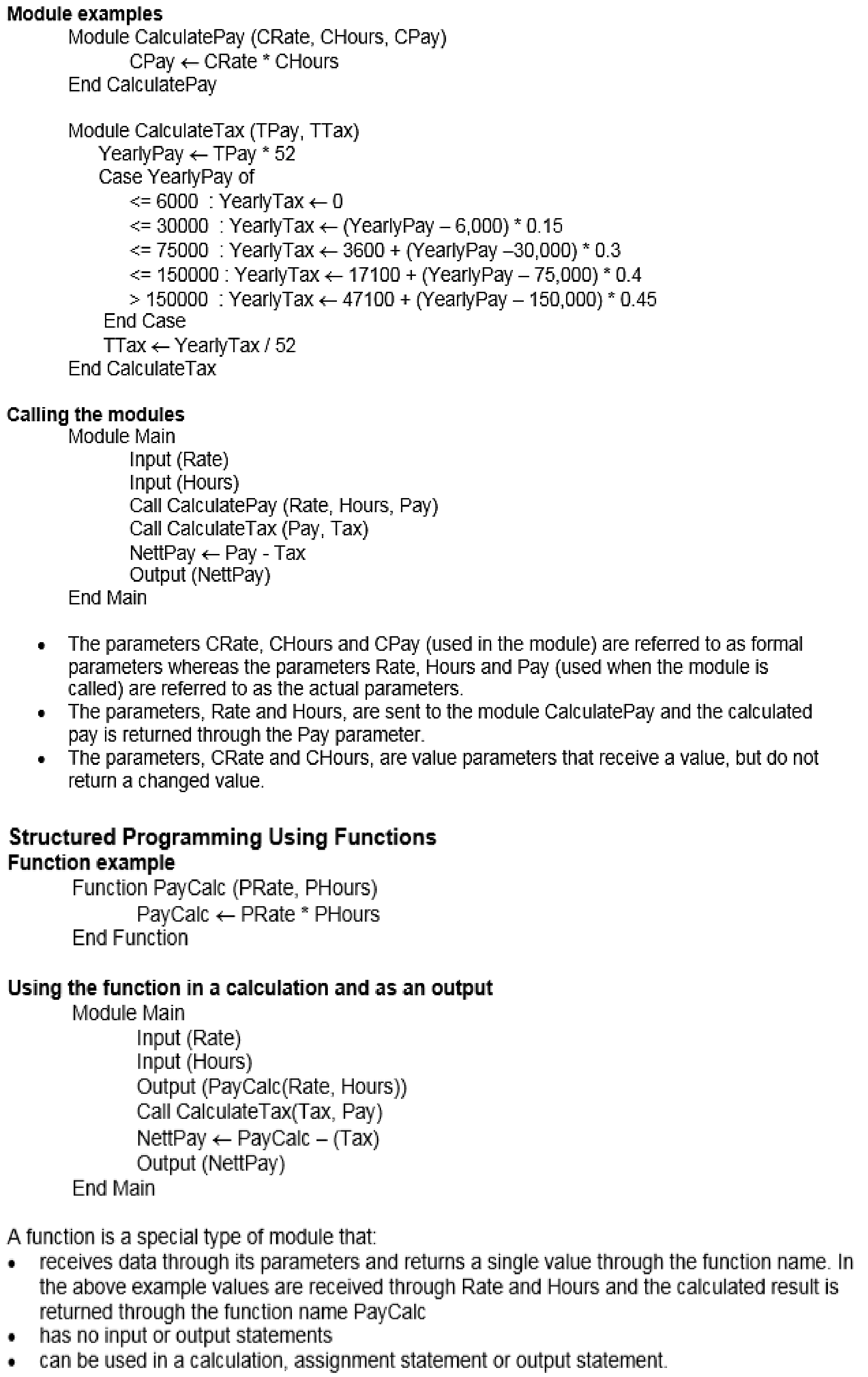
- ▪ storyboard of the intended user interface with annotations (completed)
- ▪ **pseudocode/algorithms (to be carried out at this stage)**
- ▪ desk check with a trace table
- ▪ create a flow chart
- ▪ create the program, including testing and evaluation.
- All code must be commented to receive marks and can be completed in a programming language of the student’s choice. It is recommended students use JavaScript, Python or any object-orientated language).

<table>
<tr><th>Type</th><th>Subject</th><th>Price</th></tr>
<tr><td><b>MESH</b></td><td>Mathematics, English, Science, HASS</td><td>$200</td></tr>
<tr><td><b>Electives</b></td><td>Computer Programming</td><td>$50</td></tr>
<tr><td></td><td>Wood Work</td><td>$140</td></tr>
<tr><td></td><td>Outdoor Education</td><td>$100</td></tr>
<tr><td></td><td>Art</td><td>$10</td></tr>
<tr><td></td><td>Media</td><td>$80</td></tr>
<tr><td></td><td>Cooking</td><td>$138</td></tr>
<tr><td></td><td>Accounting and Finance</td><td>$50</td></tr>
</table>

</td>
</tr>
</table>

Weeks	Syllabus content
10–11	**Digital Implementation** Design algorithms represented diagrammatically and in structured English, including iteration

Content unpacked



Module examples

```
Module CalculatePay (CRate, CHours, CPay)
    CPay ← CRate * CHours
End CalculatePay

Module CalculateTax (TPay, TTax)
  YearlyPay ← TPay * 52
  Case YearlyPay of
      <= 6000  : YearlyTax ← 0
      <= 30000  : YearlyTax ← (YearlyPay – 6,000) * 0.15
      <= 75000  : YearlyTax ← 3600 + (YearlyPay –30,000) * 0.3
      <= 150000 : YearlyTax ← 17100 + (YearlyPay – 75,000) * 0.4
      > 150000  : YearlyTax ← 47100 + (YearlyPay – 150,000) * 0.45
  End Case
  TTax ← YearlyTax / 52
End CalculateTax
```

Calling the modules

```
Module Main
    Input (Rate)
    Input (Hours)
    Call CalculatePay (Rate, Hours, Pay)
    Call CalculateTax (Pay, Tax)
    NettPay ← Pay - Tax
    Output (NettPay)
End Main
```

- The parameters CRate, CHours and CPay (used in the module) are referred to as formal parameters whereas the parameters Rate, Hours and Pay (used when the module is called) are referred to as the actual parameters.
- The parameters, Rate and Hours, are sent to the module CalculatePay and the calculated pay is returned through the Pay parameter.
- The parameters, CRate and CHours, are value parameters that receive a value, but do not return a changed value.

Structured Programming Using Functions

Function example

```
Function PayCalc (PRate, PHours)
    PayCalc ← PRate * PHours
End Function
```

Using the function in a calculation and as an output

```
Module Main
    Input (Rate)
    Input (Hours)
    Output (PayCalc(Rate, Hours))
    Call CalculateTax(Tax, Pay)
    NettPay ← PayCalc – (Tax)
    Output (NettPay)
End Main
```

A function is a special type of module that:

- receives data through its parameters and returns a single value through the function name. In the above example values are received through Rate and Hours and the calculated result is returned through the function name PayCalc
- has no input or output statements
- can be used in a calculation, assignment statement or output statement.

Suggested teaching and learning experiences

- Students create algorithms and/or flow charts for a variety of common programs, for example:
  - basic tax calculator
  - cricket score calculator
  - choosing clothes for a character
  - Pac-Man game
- Students use a traceable to validate their programs and analyse the outputs.

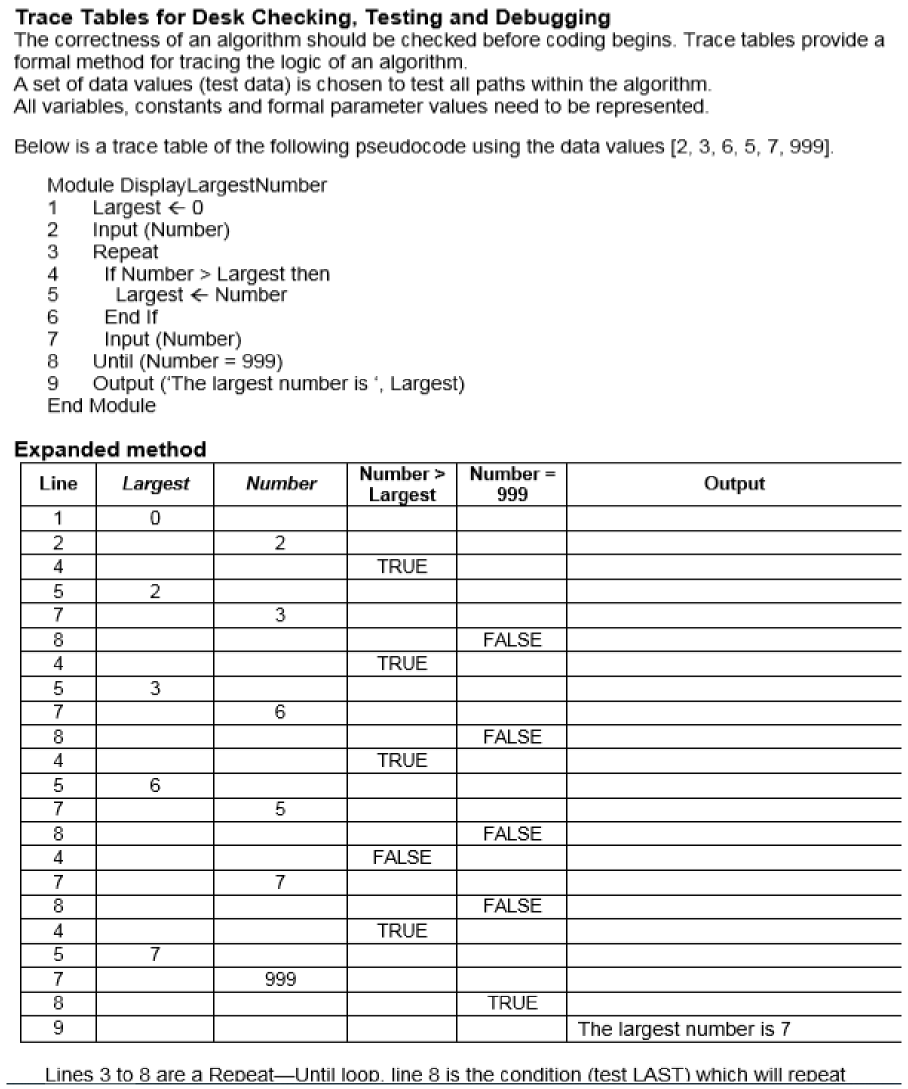
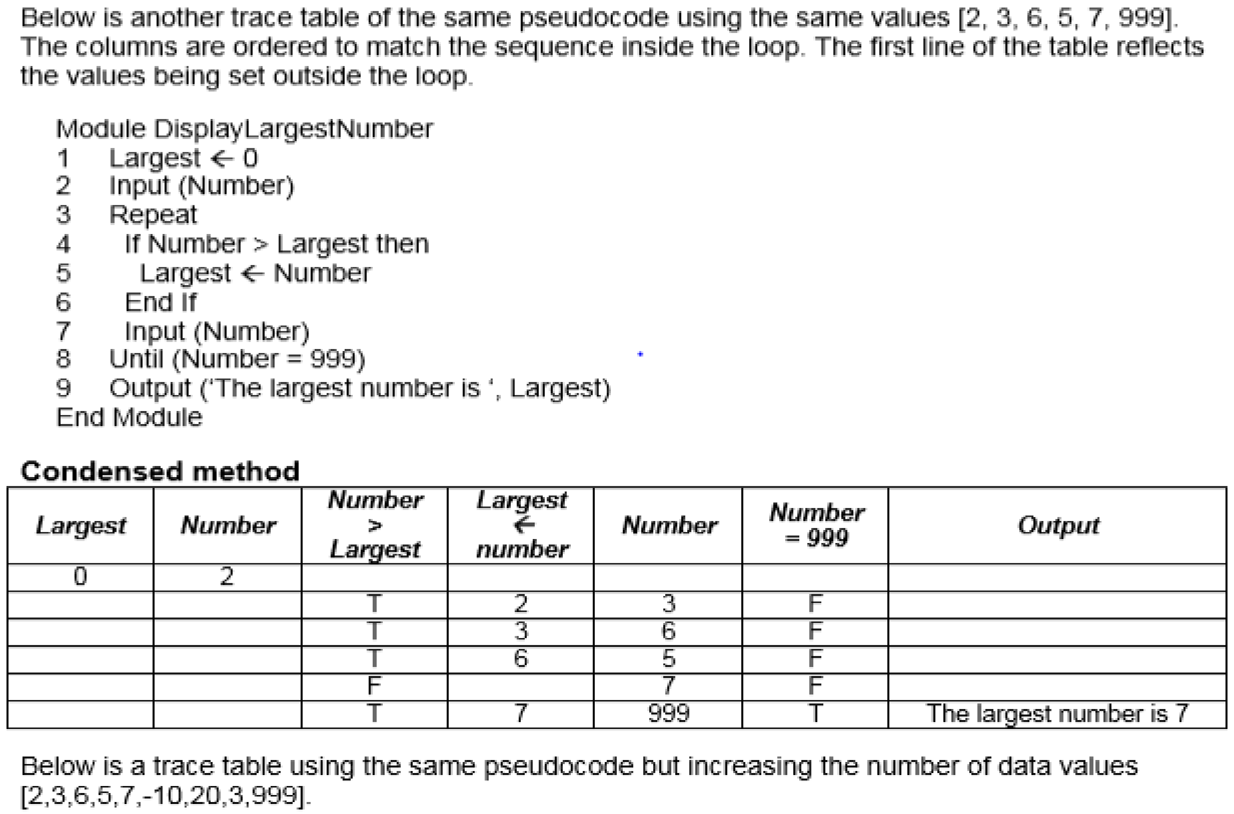
Programming – choose your own code-venture

- Your school needs your expertise in creating an online subject selection payment system. This program needs to add up the cost of student courses in order to enrol them.
- Obviously MESH subjects are compulsory (**constant**) but students can choose only two electives. Each elective has different values (**variables**).
- Students should only be able to add two of the electives, so use **data validation techniques** and **control structures** to do this.
- The Price list is attached, and must output student details as well as total price for the next year. Design of the interface is up to the student, but must be storyboarded.
- This task is broken up into 5 stages:
  a) storyboard the intended user interface with annotations (completed)
  b) **pseudocode/algorithms (to be carried out at this stage)**
  c) desk check with a trace table
  d) create flow chart
  e) creating the program including testing and evaluation.
- All code must be commented to receive marks and can be completed in a programming language of your choice. (Recommend JavaScript, Python or any object-orientated language)

Type	Subject	Price
MESH	Mathematics, English, Science, HASS	$200
Electives	Computer Programming	$50
	Wood Work	$140
	Outdoor Ed	$100
	Art	$10
	Media	$80
	Cooking	$138
	Accounting and Finance	$50

Useful links

- YouTube Intro to Algorithms: Crash Course Computer Science #13 (https://youtu.be/rL8X2mlNHPM)

Weeks	Syllabus content	Content unpacked	Suggested teaching and learning experiences
12–15	**Digital Implementation** Validate algorithms and programs using common acceptable methods	Validation of algorithms is generally completed through desk checking or tracing (Trace Table) with the intended purpose/s based on predetermined inputs. Note: alternatives ways of validating can be considered. 	 **Programming– choose your own code-venture** • Students need to create an online subject selection payment system. This program needs to add up the cost of student courses in order to enrol them. • Mathematics, English, Science and Humanities and Social Sciences (HASS) subjects are compulsory (**constant**) but students can choose only two electives. Each elective has different values (**variables**). • Students should only be able to add two of the electives, so students should use **data validation techniques** and **control structures** to do this. • Using the price list below, students must output student details as well as the total price for the next year. Design of the interface is up to the student, but must be storyboarded. • The task is broken up into five stages: ▪ storyboard of the intended user interface with annotations (completed) ▪ pseudocode/algorithms (completed) ▪ **desk check with a trace table (to be carried out at this stage)** ▪ **create a flow chart (to be carried out at this stage)** ▪ create the program, including testing and evaluation. • All code must be commented to receive marks and can be completed in a programming language of the student's choice. Recommend students use JavaScript, Python or any object-orientated language.

Trace Tables for Desk Checking, Testing and Debugging

The correctness of an algorithm should be checked before coding begins. Trace tables provide a formal method for tracing the logic of an algorithm.

A set of data values (test data) is chosen to test all paths within the algorithm.

All variables, constants and formal parameter values need to be represented.

Below is a trace table of the following pseudocode using the data values [2, 3, 6, 5, 7, 999].

```
Module DisplayLargestNumber
1   Largest ← 0
2   Input (Number)
3   Repeat
4     If Number > Largest then
5       Largest ← Number
6     End If
7     Input (Number)
8   Until (Number = 999)
9   Output ('The largest number is ', Largest)
End Module
```

Expanded method

Line	*Largest*	*Number*	Number > Largest	Number = 999	Output
1	0				
2		2			
4			TRUE		
5	2				
7		3			
8				FALSE	
4			TRUE		
5	3				
7		6			
8				FALSE	
4			TRUE		
5	6				
7		5			
8				FALSE	
4			FALSE		
7		7			
8				FALSE	
4			TRUE		
5	7				
7		999			
8				TRUE	
9					The largest number is 7

Lines 3 to 8 are a Repeat—Until loop, line 8 is the condition (test LAST) which will repeat

Below is another trace table of the same pseudocode using the same values [2, 3, 6, 5, 7, 999]. The columns are ordered to match the sequence inside the loop. The first line of the table reflects the values being set outside the loop.

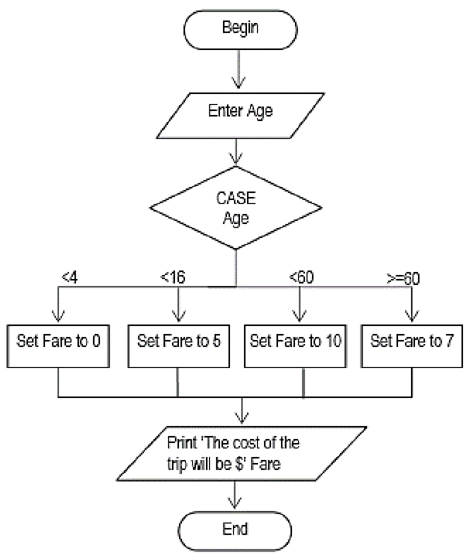
```
Module DisplayLargestNumber
1   Largest ← 0
2   Input (Number)
3   Repeat
4     If Number > Largest then
5       Largest ← Number
6     End If
7     Input (Number)
8   Until (Number = 999)
9   Output ('The largest number is ', Largest)
End Module
```

Condensed method

Largest	*Number*	*Number > Largest*	*Largest ← number*	*Number*	*Number = 999*	*Output*
0	2					
		T	2	3	F	
		T	3	6	F	
		T	6	5	F	
		F		7	F	
		T	7	999	T	The largest number is 7

Below is a trace table using the same pseudocode but increasing the number of data values [2,3,6,5,7,-10,20,3,999].

Type	Subject	Price
MESH	Mathematics, English, Science, HASS	$200
Electives	Computer Programming	$50
	Woodwork	$140
	Outdoor Education	$100
	Art	$10
	Media	$80
	Cooking	$138
	Accounting and Finance	$50

<table>
<tr><th>Weeks</th><th>Syllabus content</th><th>Content unpacked</th><th>Suggested teaching and learning experiences</th></tr>
<tr><td></td><td></td><td>Diagrammatical designs can include storyboards and flowcharts.
<table>
<tr><th>Flowchart</th><th>Pseudocode</th></tr>
<tr><td>
</td><td><b>Multi way selection (Case)</b>
Input (Age)
Case Age of
&lt; 4 : Fare ← 0
&lt; 16 : Fare ← 5
&lt; 60 : Fare ← 10
&gt;= 60 : Fare ← 7
End Case
Output ('The cost of the trip will be $', Fare)</td></tr>
</table></td><td></td></tr>
<tr><td>16–17</td><td><b>Digital Implementation</b>
Implement data storage and organisation techniques within a programming environment</td><td>• At this level, students should be creating solutions to everyday programs in an object-oriented environment. Object-orientated programming (OOP) languages focus on attaching data to objects rather than logical functions. These languages include Python, JavaScript, Java, Visual Basic, C++, Swift, Ruby etc.
• Sequence of steps refers to the order of a program. This must be logical to avoid any logic errors.
• Control structures are expressions used in programming to make decisions and iterate code.
• Branching is a term that refers to having more than one intended outcome.
• The digital solution must have choices – these are called selection. The three types of selection are:
▪ on-way selection (if-then)
▪ two-way selection (if-then-else)
▪ multiway selection (CASE)
• Iteration:
▪ test first (do while)
▪ test last (repeat loop)
▪ fixed (for loop)
• Modularity/Modularisation is the technique of writing smaller independent functions, or sections, that interact and pass parameters between the modules when required. Modularisation allows for the interchangeability of these modules when required. Modularisation is used over programming one long continuous piece of code, as it is more time efficient in having the ability of reusing code, easier to debug and isolate errors, easier to read and lends itself to collaborative team work.
• YouTube Pseudocode: Software Modularisation (https://youtu.be/DWIgTlaY1SE)</td><td><b>Activity/Assessment – choose your own code-venture</b>
• Students need to create an online subject selection payment system. This program needs to add up the cost of student courses in order to enrol them.
• Mathematics, English, Science and Humanities and Social Sciences (HASS) subjects are compulsory (<b>constant</b>) but students can choose only two electives. Each elective has different values (<b>variables</b>).
• Students should only be able to add two of the electives, so students should use <b>data validation techniques</b> and <b>control structures</b> to do this.
• Using the price list below, students must output student details as well as the total price for the next year. Design of the interface is up to the student, but must be storyboarded.
• The task is broken up into five stages:
▪ storyboard of the intended user interface with annotations (completed)
▪ pseudocode/algorithms (completed)
▪ desk check with a trace table (completed)
▪ create a flow chart (completed)
▪ <b>create the program, including testing and evaluation (to be carried out at this stage).</b>
YouTube Pseudocode: Software Modularisation (https://youtu.be/DWIgTlaY1SE)
• All code must be commented to receive marks and can be completed in a programming language of the student's choice. Recommend students use JavaScript, Python or any object-orientated language.
<table>
<tr><th>Type</th><th>Subject</th><th>Price</th></tr>
<tr><td><b>MESH</b></td><td>Mathematics, English, Science, HASS</td><td>$200</td></tr>
<tr><td><b>Electives</b></td><td>Computer Programming</td><td>$50</td></tr>
<tr><td></td><td>Woodwork</td><td>$140</td></tr>
<tr><td></td><td>Outdoor Education</td><td>$100</td></tr>
<tr><td></td><td>Art</td><td>$10</td></tr>
<tr><td></td><td>Media</td><td>$80</td></tr>
<tr><td></td><td>Cooking</td><td>$138</td></tr>
<tr><td></td><td>Accounting and Finance</td><td>$50</td></tr>
</table></td></tr>
</table>

Weeks	Syllabus content	Content unpacked	Suggested teaching and learning experiences
18	**Evaluating** Analyse design processes and solutions against student developed criteria **Digital Implementation** Create interactive solutions for sharing ideas and information online, taking into account social contexts and legal responsibilities **Collaborating and managing** Work independently, and collaboratively to manage projects, using digital technology and an iterative and collaborative approach. Considers time, cost, risk, safety, production processes, sustainability and legal responsibilities	• Evaluation should be conducted throughout the software development process to allow for reflection and future best practice. • Justification against the design process and comparison to its intended purpose is critical, not just the finished product. • Collaboration and sharing ideas online can take many forms, including Google Hangouts™, MS Teams™, Slack™, WebEx™, Asana™, Codingteam™ etc. • Legal responsibilities online refer to the following of laws: ▪ Electronic Frontiers Defamation Laws and the Internet (online defamation) (https://www.efa.org.au/Issues/Censor/defamation.html) ▪ Australian Government Office of the Australian Information Commissioner The Privacy Act (https://www.oaic.gov.au/privacy/the-privacy-act/) ▪ Australian Communications and Media Authority Avoid sending spam (*Spam Act 2003*) (https://www.acma.gov.au/avoid-sending-spam) ▪ Onyx Online Law Internet Piracy Laws Australia (piracy and copyright) (https://onyxonlinelaw.com/internet-piracy-laws-australia/).	• Teacher uses evaluation of code, including internal documentation in the form of in script commenting to explain the process and the explanation of the tags used. Students should present their source code and explain it through a presentation to the class to authenticate their knowledge. • Managing a group project: augmented reality (AR). The creation of this project should revolve around the impacts of technology or legal and social responsibilities. Useful links: • Digital Technologies Hub Managing a group project: augmented reality (https://www.digitaltechnologieshub.edu.au/teachers/scope-and-sequence/9-10/interactions-and-impacts/managing-a-group-project). • https://www.esafety.gov.au/educators/classroom-resources/yes-project • https://foundation.mozilla.org/en/initiatives/web-literacy/ • https://www.australia.gov.au/information-and-services/public-safety-and-law/online-safety • https://www.cyber.gov.au/advice • School Curriculum and Standards Authority Digital Technologies Year 10 Sample Assessment Task Topic test (https://k10outline.scsa.wa.edu.au/home/assessment/assessment-activities/year10)
19	**Representation of data** Simple compression of data and how content data is separated from presentation data	• Compression is reducing the size of a file by changing its properties. Compression aims to reduce file size without significantly reducing the quality of the file. Smaller file sizes have faster transfer speeds. Tech Target data compression (https://searchstorage.techtarget.com/definition/compression). • Two main types of compression lossy and lossless. YouTube Understanding lossy and lossless compression (https://youtu.be/2Qo5prktYNQ). ▪ Lossy compression – a file is compressed with major loss of data. This data cannot be fully restored. Creates smaller file sizes and reduced quality of image. Formats include JPEG, MPEG, WMA. ▪ Lossless compression – elimination of small elements of data to compress a file allowing it to be restored to a higher level than a lossy file. Less data is lost, resulting in larger file sizes and higher quality of image. Formats include PNG, Raw, MP4.	• Open a supplied source video in a video transcoder, such as HandBrake (https://handbrake.fr/). Students then save the supplied source video in a variety of different formats to compare and contrast their features, including file size and quality. This is easily represented in a table format. Useful links: • https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/seeing-the-big-picture • https://csfieldguide.org.nz/en/chapters/coding-compression/ • https://csfieldguide.org.nz/en/chapters/data-representation/ • https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/everything-you-always-wanted-to-know • https://australiancurriculum.edu.au/resources/work-samples/samples/digital-project-representation-of-data-below/

Note: the above Teaching and Learning Outline is based on two hours per week for 19 weeks for a total of 38 hours.