Government of Western Australia
School Curriculum and Standards Authority

COMPUTER SCIENCE

GENERAL COURSE

Year 12 syllabus

Acknowledgement of Country

Kaya. The School Curriculum and Standards Authority (the Authority) acknowledges that our offices are on Whadjuk Noongar boodjar and that we deliver our services on the country of many traditional custodians and language groups throughout Western Australia. The Authority acknowledges the traditional custodians throughout Western Australia and their continuing connection to land, waters and community. We offer our respect to Elders past and present.

Important information

This syllabus is effective from 1 January 2024.

Users of this syllabus are responsible for checking its currency.

Syllabuses are formally reviewed by the School Curriculum and Standards Authority (the Authority) on a cyclical basis, typically every five years.

Copyright

© School Curriculum and Standards Authority, 2023

This document – apart from any third-party copyright material contained in it – may be freely copied, or communicated on an intranet, for non-commercial purposes in educational institutions, provided that the School Curriculum and Standards Authority (the Authority) is acknowledged as the copyright owner, and that the Authority's moral rights are not infringed.

Copying or communication for any other purpose can be done only within the terms of the *Copyright Act 1968* or with prior written permission of the Authority. Copying or communication of any third-party copyright material can be done only within the terms of the *Copyright Act 1968* or with permission of the copyright owners.

Any content in this document that has been derived from the Australian Curriculum may be used under the terms of the Creative Commons Attribution 4.0 International licence.

Content

Rationale

The Computer Science General course focuses on the fundamental principles, concepts and skills within the field, and provides students with opportunities to develop flexibility and adaptability in the application of these in the roles of developers and users. The underpinning knowledge and skills in computer science are practically applied to the development of computer systems and software, while the connectivity between computers, peripheral devices and software used in the home, workplace and in education are examined. Students develop problem-solving abilities and technical skills as they learn how to diagnose and solve problems in the course of understanding the building blocks of computing.

In this course, the impact of technological developments on the personal, social and professional lives of individuals, businesses and communities are investigated. The ethical, moral and legal factors that influence developments in computing is explored so that students recognise the consequences of decisions made by developers and users in respect to the development and use of technology.

This course provides students with practical and technical skills that equip them to function effectively in a world where these attributes are vital for employability and daily life in a technological society. It provides a sound understanding of computing to support students pursuing further studies in related fields.

Course outcomes

The Computer Science General course is designed to facilitate achievement of the following outcomes.

Outcome 1 – Technology process

Students apply a technology process to develop computer-based systems.

In achieving this outcome, students:

- investigate ideas and generate proposals
- develop solutions that meet specifications and recognised standards
- evaluate computer-based solutions.

Outcome 2 – Knowledge and understanding of computer-based systems

Students understand the design, application and interactions of hardware and software in computer-based systems.

In achieving this outcome, students:

- understand the appropriate selection and application of computer-based system components
- understand the nature of the interactions between the elements of computer-based systems
- understand the concepts associated with computer-based systems.

Outcome 3 – Skills for computer-based systems

Students apply skills to maintain, adapt or develop computer-based systems.

In achieving this outcome, students:

- apply a range of problem-solving techniques when maintaining or developing computer-based systems
- apply a range of conventions and standards when implementing a maintenance or development solution
- apply organisational skills to identify and use appropriate hardware and software resources when maintaining or developing a computer-based system.

Outcome 4 – Computer-based systems in society

Students understand the interrelationships between the development and use of computer-based systems, the individual and society.

In achieving this outcome, students:

- understand that developers' attitudes and values affect the development of computer-based systems
- understand that users' attitudes and values affect the development and use of computer-based systems
- understand there are legal, societal and ethical impacts when computer-based systems are developed and adopted.

Organisation

This course is organised into a Year 11 syllabus and a Year 12 syllabus. The cognitive complexity of the syllabus content increases from Year 11 to Year 12.

Structure of the syllabus

The Year 12 syllabus is divided into two units which are delivered as a pair. The notional time for the pair of units is 110 class contact hours.

Unit 3 – Developing computer-based systems and producing spreadsheet and database solutions

The focus for this unit is on developing computer-based systems and producing spreadsheet and database solutions. Students are introduced to the internal, interrelating components of computer-based systems in an industry context. They examine a variety of systems, build on their spreadsheet and database skills and gain an appreciation of how these concepts and technologies are used in industry.

Unit 4 – Developing computer-based solutions and communications

The focus for this unit is on developing computer-based systems solutions and communications. Students are introduced to networking concepts, as applied to industry. Through the use of algorithms, students develop programming skills. Students create solutions exploring the ethical, legal and societal implications of industry-based applications.

Each unit includes:

- a unit description – a short description of the focus of the unit
- unit content – the content to be taught and learned.

Organisation of content

The unit content includes both theoretical aspects (Knowledge) and practical aspects (Skills).

The course is divided into five content areas.

Unit 3 is divided into two content areas:

- Systems analysis and development
- Managing data.

Unit 4 is divided into three content areas:

- Developing software
- Programming
- Networks and communications.

Systems analysis and development

The functions and technical capabilities of systems, how components are configured to form a computer system and factors which affect the design of an information system are explored. The compatibility of components, output, bandwidth considerations, and usability, security, health and safety considerations are explored. Evaluations of systems, devices or components are conducted, while acquiring computer hardware knowledge and skills.

Managing data

The distinction between data and information, including the different types of data (including text and number) and the varied representation of data within a computer, are addressed. The representation of data types, the graphical representation of data, and how data is stored using a database are also addressed.

Developing software

A systems development cycle (SDC) that includes some basic systems engineering and the application of standards is applied. How a developer's interactions with users affect the development and use of the system is investigated. Various methods of developing software systems and the problems associated with connecting systems in an increasingly global environment are addressed. The different perspectives of users and developers to the development and use of computer-based systems are explored.

Programming

The different types of programming languages (first, second, third and fourth generation, procedural, non-procedural, object-oriented and scripting languages) are investigated. The basic constructs of sequence, selection and iteration are examined. The analysing and breaking down of problems into small, self-contained units for which procedures or functions are created in a programming language are addressed. The passing of parameters to procedures, functions and modules are explored.

Networks and communications

The various structures and components of a network, including the communication media used to combine them, are examined. The convergence of technologies which involves the integration of computers and communication hardware, is investigated. Similarly, the design and creation of networks of various configurations, as well as connecting networks of different types, are investigated. The application of connectivity standards relating to networks and the internet, is addressed. Communication software models and standards; the types, purpose and use of protocols, servers and operating systems in communications; and software and the aspects to consider in network security are explored.

Resources

It is recommended that for delivery of the Computer Science General course, students have access to the following resources:

- computers with access to the internet
- peripheral devices, including:
  - scanner/photocopier/printer (multi-function device)
  - printer(s)
- applications software
  - spreadsheet software
  - word processing software
  - presentation software
  - multimedia software
  - personal communication software
  - collaborative management software
  - browser software
  - web-authoring software.

Programming language

There is no prescribed programming language for the Computer Science General course. However, to meet the assessment requirements for this syllabus, it is required that students use a programming language that enables the:

- development of a purpose-designed software solution
- design, creation, modification, testing, evaluation and documentation of programs
- writing, compiling, interpretation, testing and debugging of code
- use and development of a user interface.

For the Computer Science General syllabus, the programming language should provide the student with opportunity to:

- use control structures, including sequence, selection and iteration
- construct and use data structures
- design and implement data validation techniques
- apply modularised and structured programming methods using modularisation and parameter passing.

There is no requirement within the Computer Science General course to create a user interface, unless required for a particular programming language (e.g. PHP).

The suggested programming languages for the Computer Science General syllabus are:

- Scratch
- Alice
- macros – VBA, application specific macros, scripting languages, Unix Bash

- Applescript
- Gamefroot
- GameMaker
- GameSalad
- Visual Basic
- programmable robotic software – RCX Code, ROBOLAB, RoboMind.

Database management systems

There is no prescribed database management system for the Computer Science General course. However, to meet the assessment requirements for this syllabus, it is required that students use a database management system that enables the:

- development of a purpose-designed database solution
- design, creation, modification, testing and evaluation of a database solution
- creation of tables, queries, forms and reports
- use and development of a user interface.

The database management systems should provide the student with opportunity to:

- create a working relational database
- construct simple queries using SQL within one table.

The suggested database management system software for the Computer Science General course are:

- Microsoft Access
- MySQL
- FileMaker
- FoxPro
- Paradox
- a spreadsheet application.

Representation of the general capabilities

The general capabilities encompass the knowledge, skills, behaviours and dispositions that will assist students to live and work successfully in the twenty-first century. Teachers may find opportunities to incorporate the capabilities into the teaching and learning program for the Computer Science General course. The general capabilities are not assessed unless they are identified within the specified unit content.

Literacy

Students become literate as they develop the knowledge, skills and dispositions to interpret and use language confidently for learning and communicating in and out of school and for participating effectively in society. Literacy involves students listening to, reading, viewing, speaking, writing and creating oral, print, visual and digital texts, and using and modifying language for different purposes in a range of contexts.

In the Computer Science General course, students develop literacy capability as they learn how to communicate ideas, concepts and detailed proposals to a variety of audiences; recognise how language can be used to manipulate meaning; and read and interpret detailed written instructions. They learn to understand and use language to discuss and communicate information, concepts and ideas related to the course.

By learning the literacy of the Computer Science General course, students understand that language varies according to context and they increase their ability to use language flexibly. The Computer Science General course vocabulary is technical and includes specific terms for concepts, processes and production. Students learn to understand that much technological information is presented in the form of drawings, diagrams, flow charts, models, tables and graphs. They also learn the importance of listening, talking and discussing in technologies processes, especially in articulating, questioning and evaluating ideas.

Numeracy

Students become numerate as they develop the knowledge and skills to use mathematics confidently across other learning areas at school and in their lives more broadly. Numeracy involves students in recognising and understanding the role of mathematics in the world, and having the dispositions and capacities to use mathematical knowledge and skills purposefully.

In the Computer Science General course, students work with the concepts of number, geometry, scale and proportion. They use models, create accurate technical drawings, work with digital models and use computational thinking in decision making processes when designing and creating best-fit solutions.

Information and communication technology capability

Students develop information and communication technology (ICT) capability as they learn to use ICT effectively and appropriately to access, create and communicate information and ideas, solve problems and work collaboratively, and in their lives beyond school. The capability involves students in learning to make the most of the digital technologies available to them. They adapt to new ways of doing things as technologies evolve, and limit the risks to themselves and others in a digital environment.

In the Computer Science General course, students create solutions that consider social and environmental factors when operating digital systems with digital information. They develop an understanding of the characteristics of data, digital systems, audiences, procedures and computational thinking. They apply this when they investigate, communicate and create purpose-designed digital solutions.

Students learn to formulate problems, logically organise and analyse data, and represent it in abstract forms. They automate solutions through algorithmic logic. Students decide the best combinations of data, procedures and human and physical resources to generate efficient and effective digital solutions.

Critical and creative thinking

Students develop capability in critical and creative thinking as they learn to generate and evaluate knowledge, clarify concepts and ideas, seek possibilities, consider alternatives and solve problems. Critical and creative thinking are integral to activities that require students to think broadly and deeply using skills, behaviours and dispositions, such as reason, logic, resourcefulness, imagination and innovation in all learning areas at school and in their lives beyond school.

In the Computer Science General course, students develop capability in critical and creative thinking as they imagine, generate, develop, produce and critically evaluate ideas. They develop reasoning and the capacity for abstraction through challenging problems that do not have straightforward solutions. Students analyse problems, refine concepts and reflect on the decision-making process by engaging in systems, design and computational thinking. They identify, explore and clarify technologies, information and use that knowledge in a range of situations. In the Computer Science General course, students think critically and creatively, they consider how data and information systems impact on our lives, and how these elements might be better designed and managed.

Personal and social capability

Students develop personal and social capability as they learn to understand themselves and others, and manage their relationships, lives, work and learning more effectively. The capability involves students in a range of practices, including recognising and regulating emotions, developing empathy for others and understanding relationships, establishing and building positive relationships, making responsible decisions, working effectively in teams, handling challenging situations constructively and developing leadership skills.

In the Computer Science General course, students develop personal and social capability as they engage in project management and development in a collaborative workspace. They direct their own learning, plan and carry out investigations, and become independent learners who can apply design thinking, technologies understanding and skills when making decisions. Students develop social and employability skills through working cooperatively in teams, sharing resources, tools, equipment and processes, making group decisions, resolving conflict and showing leadership. Designing and innovation involve a degree of risk taking and as students work with the uncertainty of sharing new ideas, they develop resilience.

The Computer Science General course enhances students' personal and social capability by developing their social awareness. Students develop understanding of diversity by researching and identifying user needs. They develop social responsibility through the understanding of empathy and respect for others.

Ethical understanding

Students develop ethical understanding as they identify and investigate concepts, values, character traits and principles, and understand how reasoning can help ethical judgement. Ethical understanding involves students in building a strong personal and socially oriented, ethical outlook that helps them to manage context, conflict and uncertainty, and to develop an awareness of the influence that their values and behaviour have on others.

In the Computer Science General course, students develop the capacity to understand and apply ethical and socially responsible principles when collaborating with others and creating, sharing and using technologies data, processes, tools and equipment. In Computer Science, students consider their own roles and responsibilities as discerning citizens, and learn to detect bias and inaccuracies. Understanding the protection of data, intellectual property and individual privacy in the school environment helps students to be ethical digital citizens.

Intercultural understanding

Students develop intercultural understanding as they learn to value their own cultures, languages and beliefs, and those of others. They come to understand how personal, group and national identities are shaped, and the variable and changing nature of culture. The capability involves students in learning about and engaging with diverse cultures in ways that recognise commonalities and differences, create connections with others and cultivate mutual respect.

In the Computer Science General course, students consider how technologies are used in diverse communities at local, national, regional and global levels, including their impact and potential to transform people's lives. They explore ways in which past and present practices enable people to use technologies to interact with one another across cultural boundaries.

Representation of the cross-curriculum priorities

The cross-curriculum priorities address the contemporary issues which students face in a globalised world. Teachers may find opportunities to incorporate the priorities into the teaching and learning program for the Computer Science General course. The cross-curriculum priorities are not assessed unless they are identified within the specified unit content.

Aboriginal and Torres Strait Islander histories and cultures

The Computer Science General course may provide opportunities for students to explore creative, engaging and diverse learning contexts for students to value and appreciate the contribution by the world's oldest continuous living cultures to past, present and emerging technologies.

Asia and Australia's engagement with Asia

The Computer Science General course may provide opportunities for students to explore contemporary and emerging technological achievements that the Asia region and Pacific region have made, and continue to make, to global technological advances, including; innovation in hardware and software design and development; the regions' role in outsourcing of information and communication technologies (ICT) services; and globalisation. Students could also consider the contribution of Australia's contemporary and emerging technological achievements to the Asia and Pacific regions.

Sustainability

The Computer Science General course may provide opportunities for students, within authentic contexts, to choose and evaluate digital technologies and information systems with regard to risks and opportunities they present. They may also evaluate the extent to which digital solutions can embrace and promote sustainable practices.

Unit 3 – Developing computer-based systems and producing spreadsheet and database solutions

Unit description

The focus for this unit is on developing computer-based systems and producing spreadsheet and database solutions. Students are introduced to the internal, interrelating components of computer-based systems in an industry context. They examine a variety of systems, build on spreadsheet and database skills and gain an appreciation of how these concepts and technologies are used in industry.

Unit content

An understanding of the Year 11 content is assumed knowledge for students in Year 12. It is recommended that students studying Unit 3 and Unit 4 have completed Unit 1 and Unit 2.

This unit includes the knowledge, understandings and skills described below.

The content includes theoretical aspects (Knowledge) and practical aspects (Skills) and is organised into the following content areas:

- Systems analysis and development
- Managing data.

Typically, approximately 60 per cent of class time would be allocated for the Managing data content and approximately 40 per cent would be allocated for Systems analysis and development content.

Systems analysis and development

Knowledge

- the concept of project management, including:
  - planning
  - scheduling
  - budgeting
  - tracking
- types of system development methodologies
  - prototyping
  - system development life cycle (SDLC)
- stages of the SDLC
  - preliminary analysis
  - analysis
  - design
  - development
  - implementation
  - evaluation and maintenance
- systems development documentation as a part of the SDLC
  - context diagrams using Yourdon/DeMarco notation

- computer system hardware and software
- the concept of boot process
- storage capacities, including:
  - bit
  - byte
  - kilobyte
  - megabyte
  - gigabyte
  - terabyte
- appropriate hardware components for a computer system designed for a specific purpose, including:
  - input
  - output
  - processing
  - storage (primary and secondary)
- the role of the standard operating environment (SOE)
- functions of the components of the central processing unit (CPU)
  - arithmetic logic unit (ALU)
  - control unit (CU)
  - registers
  - program counter
  - system clock
- the concept of the fetch-execute cycle
- troubleshooting strategies, including:
  - diagnosis of fault
  - implement a solution
  - document troubleshoot procedure
- appropriate physical preventative maintenance measures
- the purpose of an ICT code of conduct
- ethics in the development and use of ICT systems
- privacy considerations in the development and use of ICT systems
- digital communications etiquette when using ICT systems

Skills

- analyse context diagrams
- document an existing system
- create context diagrams using Yourdon/DeMarco notation, as a part of the SDLC

Managing Data

Knowledge

- spreadsheet terms, including:
  - cell
  - formula
  - function (sum, average, max, min, count, countif)
  - label
  - worksheet
  - lookup tables (hlookup, vlookup)
- hierarchical structure of data
  - character/byte
  - field
  - record
  - table/relation
- data protection methods, including:
  - encryption
  - authentication
    - passwords
    - biometric
    - digital signature
- data types, including:
  - number
  - date/time
  - currency
  - text (string)
  - Boolean (true/false)
- database terms, including:
  - data, field and record
  - data integrity
  - data redundancy
- ethical and legal issues relating to in the personal use and storage of data
- legal requirements and implication of information kept by various organisations about individuals
- issues related to use of online databases
- design considerations for visual interfaces and navigation systems within database systems
- the purpose of database documentation for the user

Skills

- create solutions using a spreadsheet application using:
  - functions
  - charts
  - lookup functions
  - sorting
- create a working single table database which includes:
  - data types
  - primary keys
  - forms
  - reports
  - queries
- create a visual interface for users of a database
- create database documentation

Unit 4 – Developing computer-based solutions and communications

Unit description

This unit builds on the content covered in Unit 3. The focus for this unit is on developing computer-based systems solutions and communications. Students are introduced to networking concepts, as applied to industry. Through the use of algorithms, students develop programming skills. Students create solutions exploring the ethical, legal and societal implications of industry-based applications.

Unit content

This unit includes the knowledge, understandings and skills described below.

The content includes theoretical aspects (Knowledge) and practical aspects (Skills) and is organised into the following content areas:

- Developing software
- Programming
- Networks and communications.

Typically, approximately 60 per cent of class time would be allocated for the Programming content, approximately 20 per cent would be allocated for Developing software content, and approximately 20 per cent would be allocated for Networks and communications content.

Developing software

Knowledge

- purpose and function of software to operate a computer system
  - operating systems
  - utility software, including:
    - file compression
    - defragmenter
    - anti-virus
    - anti-malware
  - application software
- requirements for software licensing, including:
  - freeware
  - open source
  - shareware
- stages of the software development cycle (SDC)
  - state the problem
  - plan and design
  - develop the solution
  - test the solution
  - evaluate the solution

- factors affecting the development of software, including:
  - user needs
  - user interface

Skills

- apply software development requirements, including:
  - user needs
  - user interface
- apply the SDC to create a digital solution

Programming

Knowledge

- characteristics of data types, including:
  - integer
  - real (floating point number)
  - Boolean
  - character
- naming conventions for variables
- types of code, including:
  - source
  - executable
- types of control structures, including:
  - sequence
  - selection
    - one-way (if then)
    - two-way (if then else)
    - multi-way (nested if)
  - iteration
    - test first (while)
    - test last (repeat until)
    - fixed (for)
- types of program or code errors, including:
  - syntax errors
  - run-time errors
  - logical errors
- the concept of data validation, including:
  - test data
  - trace table
- modelling of an algorithm to test for logic using flow charts

Skills

- create flow charts to represent a programming solution
- use pseudocode to represent a programming solution
- apply, using pseudocode and a programming language, the following programming concepts:
  - constants
  - variables
- apply, using pseudocode and a programming language, the following control structures:
  - sequence
  - selection
  - iteration
- apply, using pseudocode and a programming language, the following techniques:
  - develop internal and external documentation
  - select and apply suitable test data for checking the solution
  - use trace tables to test for and debug logic errors
- apply the SDC to create a digital solution

Networks and communications

Knowledge

- functions of the following computer hardware components required for networks
  - router
  - switch
  - firewall
  - modem
  - network interface card (NIC)
  - wireless access point
  - bridge
- communication terms, including:
  - protocols
  - digital
  - analogue
  - ethernet
- types of communication networks
  - personal area network (PAN)
  - local area network (LAN)
  - wide area network (WAN)
- technologies appropriate for the implementation of a client/server and peer-to-peer network
- star network topology
- diagrammatic representation of network topologies for PAN, LAN and WAN

- characteristics of transmission media, including:
  - twisted pair
  - fibre optic
  - satellite
  - cellular
  - wireless
- types of communication protocols, including:
  - post office protocol 3 (POP3)
  - internet message access protocol (IMAP)
  - simple mail transfer protocol (SMTP)
- methods used to ensure security of information over the internet, including:
  - authentication
  - encryption
  - firewalls
- types of malware, including:
  - viruses
  - worms
  - trojans
  - spyware

Skills

- create network diagrams using CISCO network diagram conventions to represent network topologies for PAN and LAN

School-based assessment

The *Western Australian Certificate of Education (WACE) Manual* contains essential information on principles, policies and procedures for school-based assessment that needs to be read in conjunction with this syllabus.

Teachers design school-based assessment tasks to meet the needs of students. The table below provides details of the assessment types for the Computer Science General Year 12 syllabus and the weighting for each assessment type.

Assessment table – Year 12

Type of assessment	Weighting
Project The student is required to develop a spreadsheet and/or database and/or software system by using the system development life cycle and/or software development cycle. Students are provided with the stimulus materials on which the project is based. Stimulus material can include: diagrams; extracts from newspaper and journal articles; flow charts; trace tables; algorithms and algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and programs. Diagrams could include those related to computer system diagrams, network diagrams and context diagrams. The student is required to research ideas: implement a database and/or software system using a database management system and programming language, to develop and evaluate solutions and manage processes throughout the production.	50%
Theory test Typically include a combination of questions requiring short and extended answers. Short answer questions can be a mix of closed and open items that can be sectionalised or scaffolded. The student can be required to explain concepts, apply knowledge, analyse and/or interpret data and/or respond to stimulus materials. Stimulus material can include: diagrams; extracts from newspaper and journal articles; flow charts; trace tables; algorithms and algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and programs. Diagrams could include those related to computer system diagrams, network diagrams and context diagrams. Extended answer questions can be a mix of closed and open items that can be sectionalised or scaffolded typically with an increasing level of complexity. The student can be required to explain concepts; apply knowledge; analyse and/or interpret data, extended algorithms, databases, spreadsheets, tables and/or diagrams; and/or devise labelled diagrams, solutions (or parts or solutions); and/or respond to stimulus materials. Stimulus material can include: diagrams; extracts from newspaper and journal articles; flow charts; trace tables; algorithms and algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and programs. Diagrams could include those related to computer system diagrams, network diagrams and context diagrams.	20%
Practical test Typically consist of a set of questions requiring the use of spreadsheet software, programming language and/or a database management system. Spreadsheet skills assessed include creating spreadsheets that include functions, charts and lookup tables; and sorting data. Programming skills assessed include: writing code; and/or compiling, testing and/or debugging code. Database skills assessed include: creating fields, data types, keys for tables; queries, forms and/or reports.	15%
Externally set task A written task or item or set of items of 50 minutes duration developed by the School Curriculum and Standards Authority and administered by the school.	15%

Teachers are required to use the assessment table to develop an assessment outline for the pair of units.

The assessment outline must:

- include a set of assessment tasks
- include a general description of each task
- indicate the unit content to be assessed
- indicate a weighting for each task and each assessment type
- include the approximate timing of each task (for example, the week the task is conducted, or the issue and submission dates for an extended task).

In the assessment outline for the pair of units, each assessment type must be included at least once over the year/pair of units. The externally set task occurs in Term 2.

In addition to this advice on the number of assessments, students must complete one practical programming task and one practical database task to meet the minimum requirement in the practical test assessment type of the assessment table.

The set of assessment tasks must provide a representative sampling of the content for Unit 3 and Unit 4.

Assessment tasks not administered under test/controlled conditions require appropriate validation/authentication processes.

Externally set task

All students enrolled in the Computer Science General Year 12 course will complete the externally set task developed by the Authority. Schools are required to administer this task in Term 2 at a time prescribed by the Authority.

Externally set task design brief – Year 12

Time	50 minutes
Format	Written
	Conducted under invigilated conditions
	Typically between four and seven questions
	Questions can require students to refer to stimulus material which can include: extracts from newspaper and journal articles; flow charts; trace tables; algorithms and algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and programs. Diagrams could include those related to computer system diagrams, network diagrams and context diagrams
Content	The Authority informs schools during Term 3 of the previous year of the Unit 3 syllabus content on which the task will be based

Refer to the *WACE Manual* for further information.

Grading

Schools report student achievement in terms of the following grades:

Grade	Interpretation
A	Excellent achievement
B	High achievement
C	Satisfactory achievement
D	Limited achievement
E	Very low achievement

The teacher prepares a ranked list and assigns the student a grade for the pair of units. The grade is based on the student's overall performance as judged by reference to a set of pre-determined standards. These standards are defined by grade descriptions and annotated work samples. The grade descriptions for the Computer Science General Year 12 syllabus are provided in Appendix 1. They can also be accessed, together with annotated work samples, through the Guide to Grades link on the course page of the Authority website at www.scsa.wa.edu.au.

To be assigned a grade, a student must have had the opportunity to complete the education program, including the assessment program (unless the school accepts that there are exceptional and justifiable circumstances).

Refer to the *WACE Manual* for further information about the use of a ranked list in the process of assigning grades.

Appendix 1 – Grade descriptions Year 12

A	**Knowledge and understanding** Accurately uses computer science terminology to describe processes and concepts in context and with justification.
	Systems development processes Successfully gathers and refines appropriate data from a range of relevant sources. Analyses an existing information system using an appropriate methodology, and provides relevant recommendations for the development of a new and/or revised system with justification, reflecting system requirements and where relevant, presents appropriate alternatives. Consistently creates context diagrams to accurately represent an information system with correct use of diagrammatic conventions. Consistently applies the systems development life cycle approach and applies a project management approach to develop a functional and efficient digital solution based on system requirements.
	Data management skills Consistently creates functional spreadsheet solutions, accurately reflecting system requirements that include relevant functions, formulae, charts, lookup functions and sort functionality. Consistently creates functional database solutions, accurately reflecting system requirements, including relevant use of data types, primary keys, data entry forms, reports and queries, supported by an appropriate visual interface and documentation.
	Programming skills Consistently designs and creates accurate flow charts and pseudocode for algorithms. Consistently applies programming concepts and control structures, supported by appropriate internal and external documentation, to create a programming solution, using pseudocode, flowcharts and a programming language, that accurately reflects system requirements. Consistently designs and creates appropriate test data to accurately and efficiently validate algorithms and successfully test for and debug logic errors. Consistently applies the software development cycle to create an effective prototype digital solution, accurately reflecting system requirements.

B	**Knowledge and understanding** Accurately uses computer science terminology to describe processes and concepts in context.
	Systems development processes Gathers and refines appropriate data from a range of relevant sources. Analyses an existing information system using an appropriate methodology and provides recommendations for development of a new and/or revised system, reflecting system requirements. Creates context diagrams to accurately represent an information system with correct use of diagrammatic conventions. Consistently applies the systems development life cycle approach and a project management approach to develop a functional digital solution based on system requirements.
	Data management skills Creates functional spreadsheet solutions, reflecting system requirements that include relevant functions, formulae, charts, lookup functions and sort functionality. Creates functional database solutions, reflecting system requirements, consisting of the relevant use of data types, primary keys, data entry forms, reports and queries, supported by a visual interface and documentation.
	Programming skills Designs and creates accurate flow charts and pseudocode for algorithms. Consistently applies programming concepts and control structures, supported by appropriate internal and external documentation, to create a programming solution, using pseudocode, flowcharts and a programming language, that reflects system requirements. Designs and creates appropriate test data to accurately validate algorithms, and successfully test for and debug logic errors. Applies the software development cycle to create a prototype digital solution, reflecting system requirements.

C	**Knowledge and understanding** Uses computer science terminology to describe processes and concepts.
	Systems development processes Gathers and refines appropriate data from a limited range of relevant sources. Conducts an analysis of an existing information system using a methodology and provides recommendations for development of a new and/or revised system, reflecting system requirements. Creates context diagrams to represent an information system with correct use of diagrammatic conventions. Applies the systems development life cycle approach and a project management approach to develop a functional digital solution based on key system requirements.
	Data management skills Creates functional spreadsheet solutions, reflecting key system requirements that include relevant functions, formulae, charts, lookup functions and sort functionality. Creates functional database solutions, reflecting key system requirements, consisting of the use of data types, primary keys, data entry forms, reports and queries, supported by a visual interface and documentation.
	Programming skills Designs and creates accurate flow charts and pseudocode for algorithms. Applies programming concepts and control structures supported by internal and/or external documentation to create a programming solution, using pseudocode, flowcharts and a programming language, reflecting system requirements. Designs and creates test data to validate algorithms and successfully test for and debug logic errors. Applies the software development cycle to create a prototype digital solution, reflecting system requirements.

D	**Knowledge and understanding** Attempts to use computer science terminology to describe processes and concepts.
	Systems development processes Attempts to gather data from inappropriate sources. Conducts an unsuccessful analysis of an existing information system and provides an incomplete and/or incorrect analysis of the system requirements. Inconsistently creates context diagrams to incorrectly represent an information system with incorrect use of diagrammatic conventions. Inconsistently applies the systems development life cycle approach and attempts to use a project management approach to develop a digital solution.
	Data management skills Creates incomplete spreadsheet solutions, with inconsistent and/or incorrect use of functions, formulae and charts. Creates incomplete database solutions, with inconsistent and/or incorrect use of data types, data entry forms, reports and queries.
	Programming skills Inconsistently designs and creates flow charts and pseudocode for algorithms. Inconsistently applies programming concepts and control structures to create an incomplete programming solution using pseudocode, flowcharts and a programming language, partially reflecting system requirements. Inconsistently designs and/or creates incomplete and/or incorrect test data to validate algorithms and unsuccessfully test for logic errors. Inconsistently applies the software development cycle to create a prototype digital solution, reflecting system requirements.

E	Does not meet the requirements of a D grade and/or has completed insufficient assessment tasks to be assigned a higher grade.

Appendix 2 – Glossary

This glossary is provided to enable a common understanding of the key terms in this syllabus.

Algorithm
Instructions specifying the logic of a program.

Attribute
A name which defines a field in a database table.

Authentication
A system entry security measure based on user input, such as a digital signature, username and password, or forms of biometrics.

Bandwidth
Architecture: rate of data transfer in bits per second.
Communication: range of frequencies analogue signals can be carried over (measured in hertz).

Benchmarking
Software or hardware performance evaluated against standardised criteria.

Boolean
Database: data type exhibiting only two possible conditions – true or false.
Programming: an expression capable of generating only two possible outcomes – true or false.
Operator: AND, OR, NOT used to combine or exclude keywords in a condition.

Byte code
Object code run on a virtual machine allowing portability across multiple platforms.

Cardinality
The relationship defined between two relational database entities.

Character
Data type, that is one byte in size, able to hold a single alphanumeric entry.

Computer-aided software engineering (CASE)
Software tools use to assist in the development of software during the SDC, including code generation, and the creation of Gantt and PERT charts.

Condition
A statement or expression for which there can only be a true or false outcome.

Constant
Name of a memory location whose literal content does not change during the execution of the program.

Context diagram
Top level diagram that graphically defines the system boundary and the flow of data between the system and external entities.

Control structures
Constructs that control the flow of the program's execution; specifically, sequence, selection and iteration.

Convergence
Process of interlinking different technologies into a single device, for example, smart phones.

Data
Raw facts that represent real-world items which become information when organised. Singular – datum.

Data dictionary
Metadata that describes the attributes of data to be stored in a database.

Data duplication
Data that is physically duplicated across a database.

Data flow diagram (DFD)
Visual representation describing the flow of data through a system.

Data redundancy
Duplication of the same attributes within a table; attribute data that can be derived from other existing data.

Data type
The characteristics of data that can be stored in a cell, such as integer, real, Boolean and string.

Database
A collection of related data which allows input, editing and deletion, and can be queried for patterns, and produce reports and charts.

Documentation
Written text that accompanies software describing attributes, characteristics and/or qualities of the program, including, the code, data dictionary, user manual.

Domain name server (DNS)
The DNS translates host addresses (URL) into Internet Protocol (IP) addresses.

Dynamic host configuration protocol (DHCP)
A protocol that automatically assigns a unique (IP) address and/or subnet mask to a communication device joining a network.

Encryption
Process of encoding data via the implementation of an encryption key.

Entity
An entity is an object, thing person, group or idea about which data can be classified, collected and/or stored. Forms of entity used in this course are:

- Database: an entity represents a table in a relational database that holds data.
- System: the source or sink of the data which flows into or out of a system and over which the system has no control.

Error
Types of errors used in this course are:

- Syntax error: an error in the source code that does not meet the requirements of the specific programming grammar structure.
- Logical error: an error in the logic of an algorithm.
- Run-time error: an error occurring during the running of a program.

Executable code
Code which has been compiled into a low-level language program; for example, .exe, .com.

File transfer protocol (FTP)
Standard for transferring programs and data across a network.

Flow chart
Graphical representation of the sequence, selection and iteration flow within an algorithm.

Form
User interface for data entry, modification and query.

Function
User defined function is a sub routine designed for a specific task which receives data via parameters and returns a single value via the function name.

Gantt chart
A bar chart, emphasising time, used for scheduling projects.

Hypertext transfer protocol (HTTP)
Rules (protocols) governing the transfer of files (text, media, audio, video) across the Internet.

Identifier
User defined name of a program element, including, variables, constants and arrays.

Integrity
Relates to the accuracy and consistency of the data. Primary areas, include: referential, entity and domain.

Keys
A key is a field, used in a database as a unique identifier. The types of keys used in this course are:

- Primary key: an attribute which uniquely identifies a record in a table.
- Composite key: a primary key consisting of two or more attributes.
- Foreign key: an attribute in a table which refers back to a primary key in a related table.

Malware
Malicious software designed to covertly access a system and cause harm.

Module
A block of code which can exist and run alone or can call other modules. Examples may include the main module, functions, or procedures.

Open source software
Collaboratively created software which is licensed to include modifiable source code.

Parameter
An argument which can be passed by value or by reference to a function or procedure or module.

Procedure
A sub routine designed to perform a specific task which does not return a value.

Project
Stimuli in the format of a case study or narrative presented to students undertaking a task, assignment or exam.

Project management
The management of a temporary task with defined start and end parameters that includes planning, budgeting, quality control, and/or human resources.

Protocol
Agreed formal descriptions of rules and formats used when communication/network devices exchange data.

Prototype
A model of a system produced using the iterative method involving design, create, and evaluate. Used in contrast to a formal SDLC method.

Pseudocode
Human readable description of the steps within a program, based on the algorithm.

Query
A method of interrogating a database to extract information. Examples include QBE and SQL.

Radio frequency identification (RFID)
Low cost self-powered RF tags designed to track items, such as animals on a farm or products in a shop or factory.

Redundant array of inexpensive devices (RAID)
Storage technology that divides and replicates data among multiple device drives.

Relation
A table within a (relational) database.

Report
The result of a query provided in a formalised format.

Simple mail transfer protocol (SMTP)
Internet standard protocol for transmitting (sending) email.

Software development cycle (SDC)
The formalised development structure imposed upon the creation of software.

Software licence
A legal instrument governing the Intellectual Property rights of the software creator.

Source code
The original readable code created by the programmer before compilation.

Standard operating environment (SOE)
The specification of hardware, operating systems and application software to be holistically applied across an office or organisation.

Statement
A line of source code.

String
A sequence of characters (often in quotation marks) normally consisting of alpha-numerics, symbols and/or spaces.

Structured query language (SQL)
A (command line) database language that allows interrogation and manipulation of data using the following format:

- Select: specifies names of fields to be used in the query
- From: specifies the tables the data is contained in
- Where: specifies the criteria to be used to extract the data.

Syntax
The keywords and rules relating to a specific program language.

System
A set of elements or components that interact to accomplish a required outcome.

System boundary
An imaginary line separating the internal system from the outside elements.

Systems development life cycle (SDLC)
A linear system of defined stages, each of which requires completion before commencement of the following stage. The SDLC is costly, time consuming, highly documented and has little to no user input.

Topology
The physical or logical configuration of a network system.

Trace table
The manual testing of the logic of an algorithm.

Transmission control protocol/Internet protocol (TCP/IP)
TCP: a set of rules (protocols) used to transmit data packages across a network.
IP: a set of rules which allows the routing of data packages across a network.

Transmission media
The physical resources used to transmit data across a network, including cables or Wi-Fi.

Universal resource locator (URL)
The reference address to a web page (resource) on the internet.

Variable
Named memory location whose literal contents can change while the program is executed.