Government of Western Australia
School Curriculum and Standards Authority

COMPUTER SCIENCE

ATAR COURSE

Year 12 syllabus

IMPORTANT INFORMATION

This syllabus is effective from 1 January 2020.

Users of this syllabus are responsible for checking its currency.

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

Content

Rationale

The Computer Science ATAR course focuses on the fundamental principles, concepts and skills within the field of computing 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, and the connectivity between computers, peripheral devices and software used in the home, workplace and in education is 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 is investigated. The ethical, moral and legal factors that influence developments in computing are 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 ATAR 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 – Design and development of computer-based systems and database solutions

In this unit, students understand the design concepts and tools used to develop relational database systems. They consider the complex interactions between users, developers, the law, ethics and society when computer systems are used and developed.

Unit 4 – Design and development of communication systems and software solutions

In this unit, students gain the knowledge and skills to create software. They use algorithms and structured programming to design and implement software solutions for a range of problems using the Software Development Cycle. Students examine attitudes and values that lead to the creation and use of computer-based systems and their effect on society. Students consider networks, communication systems, including security and protocols.

Each unit includes:

- a unit description – a short description of the purpose 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, is addressed. The representation of data types in data dictionaries, the graphical representation of data, how data is stored into separate entities using a relational database and the process of normalisation are examined.

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 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 is addressed. The passing of parameters to procedures, functions and modules are explored. This includes the means by which records, files and databases in an application are accessed and an understanding of the operation of compilers and interpreters is developed.

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 Computer Science, students have access to the following resources:

- computers with access to the internet
- peripheral devices, including:
  - scanner/photocopier/printer (multi-function device)
  - printers

- applications software
  - spreadsheet software
  - word processing software
  - presentation software
  - multimedia software
  - personal communication software
  - collaborative management software
  - browser software
  - web-authoring software

Programming languages

There is no prescribed programming language for the Computer Science ATAR 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 ATAR course, the programming language should provide the student with opportunity to:

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

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

The suggested programming languages for the Computer Science ATAR course are:

- Visual Basic
- Pascal
- Python
- PHP
- Java
- C#
- Javascript.

Database management systems

There is no prescribed database management system for the Computer Science ATAR 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 or two tables
- construct queries across multiple tables using a database tool
- apply programmed control structures
- develop and use a user interface.

The suggested database management system software for the Computer Science ATAR course is:

- Microsoft Access
- MySQL
- FileMaker
- FoxPro
- Paradox.

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 Computer Science. 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 ATAR 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 computer science, students understand that language varies according to context and they increase their ability to use language flexibly. Computer science vocabulary is often 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 and talking when learning about 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 ATAR 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 ATAR 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 ATAR 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 ATAR 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 ATAR 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 ATAR 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 with 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 ATAR 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 this course, 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 ATAR 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 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 ATAR 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 ATAR 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 ATAR 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 communications technology (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 ATAR 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 – Design and development of computer-based systems and database solutions

Unit description

Students learn about the design concepts and tools used to develop relational database systems. They consider the complex interactions between users, developers, the law, ethics and society when computer systems are used and developed.

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. This is the examinable content.

The unit content includes theoretical aspects (Knowledge) and practical aspects (Skills) and organised into two content areas:

- Systems analysis and development
- Managing data.

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

Systems analysis and development

Knowledge

- types of system development methodologies
  - linear – waterfall/cascade
  - iterative – rapid application development (RAD)
  - advantages and disadvantages of linear and iterative system development methodologies
- stages of the system development life cycle (SDLC)
  - preliminary analysis
    - problem definition
    - feasibility study
  - analysis
    - model of current system
    - requirements of new system
  - design
    - logical and physical design
  - development
    - hardware and software acquisition
    - construction and testing
  - implementation
    - change-over methods, including: direct cut, phased, pilot and parallel

    - evaluation and maintenance
        - performance evaluation
        - fault finding and correction
- data gathering techniques used in the SDLC, including: observation, questionnaire, interview, sample forms, and sampling volume of work processed by system
- project management computer aided software engineering (CASE) tools
    - Gantt charts
    - program evaluation review technique (PERT) charts
- systems development documentation as a part of the SDLC
    - context diagrams using Yourdon/DeMarco notation
    - data flow diagrams using Yourdon/DeMarco notation
    - system manuals
    - user manuals

Skills

- apply data gathering techniques and CASE tools
- analyse user and system documentation, including: Gantt charts, PERT charts, context and data flow diagrams
- create user and system documentation as a part of the SDLC
- apply context diagrams and data flow diagrams, using Yourdon/DeMarco notation, as a part of the SDLC
    - detect errors in diagrams
    - define system boundaries
    - create accurate diagrams
    - create context diagrams
    - create Level 0 DFDs
    - create Level 1 DFDs

Knowledge

- appropriate hardware components for a computer system designed for a specific purpose
- purpose of a standard operating environment (SOE)
- advantages and disadvantages of a SOE
- roles of an operating system
    - scheduling
    - managing concurrency
    - managing memory
    - managing devices
- role of file systems
- features of file systems, including:
    - space management
    - filenames
    - directories

- role of drivers
- types of operating systems
  - embedded
  - stand alone
  - server
- role of the following components of the central processing unit (CPU)
  - arithmetic logic unit (ALU)
  - control unit (CU)
  - registers
  - program counter
  - system clock
  - data, address and control bus
- purpose of the fetch-execute cycle
- stages of the fetch-execute cycle
  - fetch
  - decode
  - execute
  - store
- purpose of processor architectures for different types of systems
- types of processing
  - distributed
  - sequential
  - parallel
  - multi-core
- purpose of using benchmarking to determine system performance:
  - software
  - hardware
  - operating systems
- purpose of disaster recovery plans
- types of disaster recovery tools, including:
  - online storage
  - incremental backup
  - full backup
  - RAID (Level 0, 1, 10)
  - uninterruptible power supply (UPS)
- benefits of virtualisation
- types of platform virtualisation
  - desktop virtualisation
  - personal computer virtualisation
  - server virtualisation
  - storage virtualisation

- purpose of platform virtualisation with application virtualisation
- purpose of cloud computing
- advantages and disadvantages of cloud computing
- convergence of technologies, including the continued development of mobile devices
- environmental issues related to the disposal of computer components
- methods for the secure disposal of data, including:
  - physical destruction of media
  - overwriting
- purpose of intellectual property in the development of ICT systems
- role of law and ethics in the use of ICT systems, including:
  - use of code of conduct policies
  - prevention of software and information piracy

Managing data

Knowledge

- types of physical storage of databases
  - online
  - local
- types of databases
  - distributed
  - centralised
- structure of data warehouses and data marts
- role of data mining
- compare data warehouses and data marts as methods of data storage and distribution
- ethical implications of the use of data warehouses, data marts and data mining
- purpose of a data dictionary
- elements of a data dictionary, including: element name, data type, size/format default, description, constraint
- database management system concepts, including:
  - data definition
  - data duplication
  - data integrity, including: referential integrity, domain integrity and entity integrity
  - data redundancy
  - data anomalies, including: insert, delete and update
  - data manipulation
  - data security
- normalisation of data to 3rd normal form (NF)

- role of open systems in:
  - database interconnectivity
  - database development
  - database management
  - data driven websites
- data modelling using Chen's notation entity relationship (ER) diagrams
- purpose of database documentation for the user
- role of law and ethics in the storage and disposal of personal data, including: the impact of privacy laws in Australia on the storage and distribution of data
- design considerations for visual interfaces and navigation systems within database systems, including:
  - readability
  - navigation
  - logical order
  - inclusivity

Skills

- analyse existing ER diagrams
- create accurate ER diagrams
- create a model of a database solution using Chen's notation entity relationship (ER) diagrams
- create data dictionaries
- create visual interface and navigation systems to assist users of a database
- create database documentation for the user
- normalise data to 3rd NF
- resolve complex many to many (M:N) relationships in a multi-table relational database system (three or more entities)
- create a working relational multi-table database using:
  - data types
  - relations
  - primary, composite and foreign keys
  - referential integrity
  - relationships, including: set cascade inserts, updates and deletes
  - cardinality (1:1, 1:M, M:1, M:N)
  - validation rules
  - forms
  - reports
  - simple queries using SQL (up to two tables), including insert, update and select queries
  - queries across multiple tables using appropriate database tools, including the following: parameter, calculated field, concatenated field, aggregation, update, delete and make table
- apply simple programmed control structures, including IF statements and calculations within the database

Unit 4 – Design and development of communication systems and software solutions

Unit description

Students gain the knowledge and skills to create software. They use algorithms and structured programming to design and implement software solutions for a range of problems using the software development cycle (SDC). Students examine attitudes and values that lead to the creation and use of computer-based systems and their effect on society. They consider networks, communication systems, including security and protocols.

Unit content

This unit builds on the content covered in Unit 3.

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

The unit content includes theoretical aspects (Knowledge) and practical aspects (Skills) organised into three content areas:

- Developing software
- Programming
- Networks and communications.

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

Developing software

Knowledge

- types of software licence requirements, including:
  - network (per seat)
  - enterprise
  - commercial/proprietary
  - end user licence agreement (EULA)
- factors affecting the development of software, including:
  - user needs
  - user interface
  - processing efficiency
  - development time
  - technical specifications
- professional ethics of developers when creating new software
- legal obligations of developers when creating new software
- legal and ethical responsibilities of software users

- stages of the software development cycle (SDC)
  - analyse detailed requirements
  - design data and algorithms
  - code data structures and instructions
  - debug syntax and logic errors
  - test to meet specifications
  - document internally and externally
  - implement and test with live data
  - evaluate performance of the program

Skills

- apply the SDC in planning and developing software solutions

Programming

Knowledge

- characteristics of simple data types:
  - integer
  - real (floating point number)
  - Boolean
  - character
- characteristics of complex data types:
  - string
  - one-dimensional arrays
  - records
- programming concepts, including:
  - constants
  - variables (local, global, parameters)
  - appropriate naming conventions for variables
  - control structures
  - stubs
  - statements
  - modularisation
  - functions
  - scope and lifetime of identifiers, including: parameter passing (value and reference)
- difference between source code, byte code and executable code
- difference between an interpreter and a compiler
- types of program or code errors, including:
  - syntax errors
  - run-time errors
  - logical errors
- purpose and characteristics of internal and external documentation

- software development documentation as a part of the SDC
  - structure charts using the Yourdon and Constantine method
- modelling of an algorithm using trace tables to test for logic
- types of data validation techniques, including:
  - range checking
  - type checking

Skills

- use pseudocode to represent a programming solution
- apply, using pseudocode and a programming language, the following programming concepts:
  - constants
  - variables (local, global, parameters)
  - naming conventions for variables
  - stubs
  - statements
  - modularisation
  - functions
  - parameter passing (value and reference)
  - one-dimensional arrays
- apply, using pseudocode and a programming language, the following control structures:
  - sequence
  - selection
    - one-way (if then)
    - two-way (if then else)
    - multi-way (case, nested if)
  - iteration
    - test first (while)
    - test last (repeat until)
    - fixed (for)
- resolve program and code errors
- apply the following algorithmic and programming techniques:
  - develop internal and external documentation
  - interpret and create structure charts with parameter passing
  - select and apply suitable test data and testing techniques
    - use trace tables to test for and debug logic errors
    - use data validation techniques, including range checking and type checking
- apply the SDC to create prototype digital solutions using pseudocode and an appropriate programming language

Networks and communications

Knowledge

- role of the following hardware devices in network communications:
  - router
  - switch
  - firewall
  - modem
  - network interface card (NIC)
  - wireless access point
  - bridge
  - gateway
  - repeaters
- role of the layers within the Department of Defence (DoD) transmission control protocol/internet protocol (TCP/IP) model in network communications
- purpose of the layers within the DoD TCP/IP model, including:
  - application layer
  - transport layer
  - internet layer
  - network layer
- characteristics of wireless transmission media, including:
  - broadcast radio
  - satellite
  - microwave
  - cellular
- characteristics of wired transmission media, including:
  - twisted pair (unshielded twisted pair [UTP] and shielded twisted pair [STP])
  - fibre optic (single-mode, multi-mode)
- similarities and differences of the carrier sense multiple access with collision detection (CSMA/CD) and the carrier sense multiple access with collision avoidance (CSMA/CA) network control protocols
- methods of error detection and correction in digital data transmission, including:
  - parity bit
  - checksum
- types of communications protocols and standards, including:
  - wireless (Bluetooth, ethernet 802.11x, radio frequency identification **[**RFID])
  - wired (ethernet 802.3)
  - TCP/IP (IP4, IP6)
  - dynamic host configuration protocol **(**DHCP) and domain name system (DNS)
- role of storage area networks (SAN) and network attached storage (NAS)
- similarities and differences of SAN and NAS

- methods used to ensure the security of networks, including use of:
  - firewalls
  - anti-virus software
  - password and network user policies
  - authentication
  - encryption
- strategies used to compromise the security of networks, including:
  - denial of service
  - back doors
  - IP spoofing
  - phishing
- factors influencing the performance of a network, including:
  - bandwidth
  - network design
  - data collisions
  - excess broadcast traffic

Skills

- create network diagrams using the CISCO network diagrammatic conventions to represent network topologies for LAN, WLAN and WAN

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 ATAR 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 database and/or software system by using the system development life cycle and/or software development cycle. Students are provided with stimulus material on which the project is based. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; Gantt and PERT charts; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or logical data flow diagrams (Level 0 and/or Level 1). The student is required to research ideas; implement a database and/or software system using a database management system and programming language; explore, develop and evaluate solutions; and manage processes throughout production to produce solutions.	30%
Theory test Tests typically consist of a combination of questions requiring short and extended answers. Short answer questions can be a mix of closed and open items that can be scaffolded or sectionalised. The student can be required to: explain concepts, apply knowledge, analyse and/or interpret data and/or refer to stimulus material. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; Gantt and PERT charts; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or logical data flow diagrams (Level 0 and/or Level 1). Extended answer questions can be a mix of closed and open items that can be scaffolded or sectionalised typically with an increasing level of complexity. The student can be required to apply knowledge and/or critical thinking skills; analyse and/or interpret data, extended algorithms, relational databases, tables and/or diagrams; devise labelled diagrams, and/or solutions (or parts of solutions). Some questions can require the student to refer to stimulus material. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; Gantt and PERT charts; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or logical data flow diagrams (Level 0 and/or Level 1).	20%
Practical test Tests typically consist of a set of questions requiring the use of a programming language and/or a relational database management system. Programming skills assessed include: writing code; and/or compiling, testing and/or debugging program code. Database skills assessed include: creating fields, data types, keys for tables, queries, forms and/or reports.	10%
Examination Typically conducted at the end of each semester and/or unit and reflecting the examination design brief for this syllabus.	40%

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 twice with the exception of Project, which could be a combined task based on both units.

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.

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 ATAR 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.

ATAR course examination

All students enrolled in the Computer Science ATAR Year 12 course are required to sit the ATAR course examination. The examination is based on a representative sampling of the content for Unit 3 and Unit 4. Details of the ATAR course examination are prescribed in the examination design brief on the following page.

Refer to the WACE Manual for further information.

Examination design brief – Year 12

Time allowed
Reading time before commencing work: ten minutes
Working time for paper: three hours

Permissible items
Standard items: pens (blue/black preferred), pencils (including coloured), sharpener, correction fluid/tape, eraser, ruler, highlighters
Special items: non-programmable calculators approved for use in the ATAR course examinations, MATHOMAT and/or Mathaid and/or any system flowchart template

Provided by the supervisor
A source booklet containing stimulus material

SECTION	SUPPORTING INFORMATION
Section One **Short answer** 40% of the total examination 20–30 questions Suggested working time: 70 minutes	Questions can be a mix of closed and open items that can be scaffolded or sectionalised. The candidate can be required to: explain concepts, apply knowledge, analyse and/or interpret data. Some questions can require the candidate to refer to stimulus material. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; Gantt and PERT charts; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or logical data flow diagrams (Level 0 and/or Level 1).
Section Two **Extended answer** 60% of the total examination 4–6 questions Suggested working time: 110 minutes	Questions can be a mix of closed and open items that can be scaffolded or sectionalised typically with an increasing level of complexity. The candidate can be required to: explain concepts; apply critical thinking skills; analyse and/or interpret complex data, extended algorithms, relational databases, tables and/or diagrams; devise labelled diagrams, and/or solutions (or parts of solutions); and/or refer to a case study. Some questions can require the candidate to refer to stimulus materials. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; Gantt and PERT charts; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or logical data flow diagrams (Level 0 and/or Level 1).

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 Consistently conducts an accurate analysis of an existing information system using appropriate data gathering techniques to develop an accurate model of the information system. Identifies the requirements, and provides detailed specifications for a proposed information system, based on analysis of an existing information system. Effectively manages the development of an information system, efficiently using appropriate CASE tools. Consistently creates Context, Level 0 and Level 1 Data Flow Diagrams to accurately represent an information system with correct use of diagrammatic conventions. Consistently applies the system development life cycle approach to develop a functional and efficient digital solution based on system requirements.
	Data management skills Consistently creates entity relationship diagrams accurately reflecting system requirements, and accurately representing relationships, cardinality, attributes, keys and diagrammatic conventions. Consistently applies normalisation to the 3rd normal form to accurately model a complex database solution. Consistently creates functional multi-table relational databases, accurately reflecting relevant system requirements and consisting of appropriate relationships, keys and data types with queries across multiple tables to efficiently extract, insert and update relevant data with data validation. Consistently creates functional and efficient queries across multiple tables, including use of parameters, calculated fields, concatenated fields and aggregation.
	Programming skills Consistently designs and creates accurate and efficient pseudocode and structure charts for algorithms, including modularisation with the correct use of functions and parameter passing. Consistently applies programming concepts and control structures to accurately create an effective programming solution using pseudocode and a programming language, accurately reflecting system requirements. Develops programming solutions that include appropriate and consistent use of functions, modularisation, local and global variables and parameter passing. Consistently designs and creates appropriate test data to accurately and efficiently validate algorithms, using trace tables and appropriate data validation techniques. 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 Conducts an analysis of an existing information system using appropriate data gathering techniques to develop a model of the information system. Identifies the key system requirements, and provides specifications for a proposed information system, based on the analysis of an existing information system. Manages the development of an information system using appropriate CASE tools. Creates Context, Level 0 and Level 1 Data Flow Diagrams to represent an information system with correct use of diagrammatic conventions. Applies the system development life cycle approach to develop a functional digital solution based on key system requirements.
	Data management skills Creates entity relationship diagrams reflecting system requirements, accurately representing relationships, cardinality, attributes, keys and diagrammatic conventions. Applies normalisation to the 3rd normal form to accurately model a simple database solution. Creates functional multi-table relational databases reflecting relevant system requirements and consisting of appropriate relationships, keys and data types with queries across multiple tables to extract, insert and update relevant data with data validation. Creates queries across multiple tables, including use of parameters, calculated fields, concatenated fields and aggregation.
	P**rogramming skills** Designs and creates accurate pseudocode and structure charts using modularisation, with the correct use of functions and parameter passing. Applies programming concepts and control structures to create a programming solution using pseudocode and a programming language, reflecting system requirements. Develops programming solutions that include use of functions, modularisation, local and global variables and parameter passing. Designs and creates test data to accurately validate algorithms, using trace tables and appropriate data validation techniques. Applies the software development cycle to create a prototype digital solution, reflecting system requirements.

C	**Knowledge and understanding** Uses computer science terminology, and describes processes and concepts.
	Systems development processes Conducts an analysis of an existing information system using a limited range of data gathering techniques to develop a model of the information system. Identifies the requirements, and provides limited specifications, for a proposed information system based on the analysis of an existing information system. Manages the development of an information system using a limited range of CASE tools. Creates Context, Level 0 and Level 1 Data Flow Diagrams to represent key aspects of an information system with minor errors in logic and/or conventions. Applies the system development life cycle approach to develop a digital solution based on limited system requirements.
	Data management skills Creates entity relationship diagrams reflecting system requirements, inconsistently representing relationships, cardinality, attributes, keys and/or diagrammatic conventions. Applies normalisation to the 3rd normal form to model a simple database solution. Creates multi-table relational databases, reflecting key aspects of system requirements and consisting of appropriate relationships, keys and data types with simple queries to extract insert and update relevant data. Creates queries across multiple tables, with use of parameters and calculated fields.
	Programming skills Designs and creates pseudocode and structure charts with a limited use of modularisation and inconsistent use of functions and parameter passing. Applies programming concepts and control structures to create a programming solution using pseudocode and a programming language, with minor errors in logic, code and/or conventions, reflecting key aspects of system requirements. Develops programming solutions that include limited use of functions, modularisation and parameter passing. Designs and creates test data to validate algorithms, using trace tables and data validation techniques.

D	**Knowledge and understanding** Attempts to use computer science terminology to describe processes and concepts.
	Systems development processes Attempts to conduct a limited analysis of an existing information system using a limited range of data gathering techniques to develop an incomplete model of the information system. Attempts to identify the requirements, and provides inappropriate and/or incomplete specifications, for a proposed information system based on the analysis of an existing system. Unsuccessfully manages the development of an information system. Creates incomplete Context, Level 0 and Level 1 Data Flow Diagrams, inaccurately representing aspects of an existing and/or proposed information system. Inconsistently and/or partially applies the system development life cycle approach to develop an incomplete digital solution.
	Data management skills Creates incomplete entity relationship diagrams partially reflecting system requirements, including main entities and attributes but with inaccurate key fields and relationships. Attempts to apply normalisation, incompletely modelling a simple database solution. Creates databases with limited functionality reflecting some aspects of system requirements. Unsuccessfully creates queries across multiple tables.
	Programming skills Inconsistently designs and creates pseudocode and/or structure charts representing incomplete and/or inaccurate algorithms. Inconsistently applies programming concepts and some control structures to create an incomplete programming solution using pseudocode and a programming language, with errors in logic, code and/or conventions, reflecting limited aspects of system requirements. Designs and creates inappropriate test data to unsuccessfully validate algorithms, and attempts to create trace tables and/or apply data validation techniques. Inconsistently applies the software development cycle, partially creating a prototype digital solution that reflects limited aspects of the 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 that specify the logic of a program.
Atomicity	Field contents which cannot be further logically divided into other fields.
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 a 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 anomaly	Data inconsistencies (update, deletion and insertion) that affect the integrity of the database.
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 in 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	Database: an entity represents a table in a relational database. System: the source or sink of the data which flows into or out of a system and over which the system has no control.
Entity relationship (ER) diagram	A data modelling technique allowing the graphical representation of the relationships between the entities in the database.
Error	Syntax: an error in the source code that does not meet the requirements of the specific programming grammar structure. Logical: an error in the logic of an algorithm. Run-time: 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	Primary key: an attribute which uniquely identifies a record in a table. Foreign key: an attribute in a table which refers back to a primary key in a related table. Composite key: a primary key consisting of two or more attributes.
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.
Normalisation	The process of identifying and eliminating data anomalies, duplication and redundancies, thereby improving data integrity and storage in a relational database.
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.
Relationship	In a relational database, the relationship describes how two tables are related to each other via the use of primary and foreign keys.
Report	The result of a query provided in a formalised format.
Resolving	The process of converting a M:N cardinality into a 1:M and M:1 set of cardinalities.
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 a 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.
Structure chart	Graphical representation of the flow of parameters between modules and functions.
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.
Stub	A simple program routine that stands in for a more complex routine to be written at a later date.

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 with 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.