Government of Western Australia
School Curriculum and Standards Authority

COMPUTER SCIENCE

ATAR COURSE

Year 11 syllabus

IMPORTANT INFORMATION

This syllabus is effective from 1 January 2015.

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 11 syllabus is divided into two units, each of one semester duration, which are typically delivered as a pair. The notional time for each unit is 55 class contact hours.

Unit 1 – Developing computer-based systems and producing spreadsheet and database solutions

The focus for this unit is 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 2 – Developing computer-based systems solutions and communications

The focus for this unit is 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. They create solutions exploring the ethical, legal and societal implications of industry-based applications.

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 1 is divided into two content areas:

- Systems analysis and development
- Managing data.

Unit 2 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, 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 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 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 the Computer Science ATAR course, 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 language

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 this 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 the Computer Science ATAR 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 are:

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

Progression from the Year 7–10 curriculum

This syllabus continues to develop student learning around the knowledge, understandings and skills within the Year 7–10 Digital Technologies curriculum and focuses on the components of digital systems (software, hardware and networks) and their use; the representation of data; and how data are represented and structured symbolically.

This syllabus also continues to develop the students' skills with the production of digital solutions through; collecting, managing and analysing data, defining problems, designing solutions, implementing and evaluating solutions, and communicating, collaborating and managing projects.

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 ATAR 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 use and interpret 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, socially oriented, and 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 when creating, sharing and using technologies data, processes, tools and equipment. In the Computer Science ATAR 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 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 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 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 1 – Developing computer-based systems and producing spreadsheet and database solutions

Unit description

The focus for this unit is 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

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

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

- 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
  - data flow diagrams (DFD) using Yourdon/DeMarco notation

Skills

- analyse context diagrams and data flow diagrams
- apply context and data flow diagrams using Yourdon/DeMarco notation, as a part of the SDLC
  - create context diagrams
  - create Level 0 DFD's (maximum four processes), applying correct symbols and rules and defining system boundaries

Knowledge

- types of primary and secondary storage, including:
  - random access memory (RAM)
  - read only memory (ROM)
  - cache
  - mechanical disk
  - solid state drive (SSD)
- concept of boot process (power up to OS booting, including POST)
- storage capacities, including:
  - bit
  - byte
  - kilobyte
  - megabyte
  - gigabyte
  - terabyte
- hardware and software components for a computer system designed for a specific purpose, including:
  - input
  - output
  - processing
  - storage (primary and secondary)
- role of standard operating environment (SOE)
- role of components in the central processing unit (CPU)
  - arithmetic logic unit (ALU)
  - control unit (CU)
  - registers
  - program counter
  - system clock
  - data, address and control bus
- concept of the fetch-execute cycle
- troubleshooting strategies, including:
  - diagnosis of fault
  - implement a solution
  - document troubleshooting procedure
- appropriate physical preventative maintenance measures

- purpose of an ICT code of conduct
- ethics in the development and use of ICT systems
- piracy considerations in the development and use of ICT systems
- digital communications etiquette when using ICT systems

Managing data

Knowledge

- spreadsheet terms, including:
  - cell
  - formula
  - label
  - functions (sum, average, max, min, count, countif)
  - worksheet
  - lookup tables (hlookup, vlookup)

Skills

- create solutions using a spreadsheet application using:
  - functions
  - charts
  - lookup table
  - sorting

Knowledge

- hierarchical structure of data
  - character/byte
  - field/attribute
  - record/tuple
  - table/entity/relation
- data protection methods, including:
  - encryption
    - private key
    - public key
  - authentication
    - passwords
    - biometrics
    - digital signature
- data types, including:
  - number
  - date/time
  - currency
  - text (string)
  - Boolean (true/false)

- database terms
  - data, field, record, relation, atomicity
  - data integrity
  - data redundancy
- ethical and legal issues relating to the personal use and storage of data
- legal requirement and implication of information kept by various organisations about individuals
- design considerations for visual interfaces and navigation systems within database systems
- purpose of database documentation for the user
- data modelling using Chen's notation entity relationships diagrams

Skills

- resolve simple many to many (M:N) relationship in a multi-table relational database system (maximum three entities)
- create using Chen's notation entity relationship (ER) diagrams for a simple database solution (maximum three entities)
- create a working relational multi-table database using:
  - data types
  - relations
  - primary and foreign keys
  - relationships
  - cardinality (1:1, 1:M, M:1, M:N)
  - validation rules
  - forms
  - reports
  - queries
- create a visual interface for a database
- create database documentation

Unit 2 – Developing computer-based systems solutions and communications

Unit description

The focus for this unit is 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. They create solutions exploring the ethical, legal and societal implications of industry-based applications.

Unit content

This unit includes knowledge, understandings and skills to the degree of complexity described below. 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

- evolution of programming languages
  - machine
  - assembler
  - procedural
  - non-procedural
  - object-oriented
- purpose and function of software to operate a computer system
  - operating systems
  - utility software
    - 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
  - test
  - evaluate
- factors affecting the development of software, including:
  - user needs
  - user interface

Skills

- create a system solution using the SDC
- apply software development requirements

Programming

Knowledge

- characteristics of data types
  - integer
  - real (floating point number)
  - Boolean
  - character
- appropriate 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 (case, 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
- modelling of an algorithm using trace tables to test for logic
- purpose of internal and external documentation
- modelling of an algorithm 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
    - one-way (if then)
    - two-way (if then else)
    - multi-way (case, nested if)
  - iteration
    - test first (while)
    - test last (repeat until)
    - fixed (for)
- 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 digital solutions
- use of the following number systems within a computer
  - binary
  - decimal
  - hexadecimal

Networks and communications

Knowledge

- functions of the following computer hardware components required for industry networks:
  - router
  - switch
  - firewall
  - modem
  - network interface card (NIC)
  - wireless access point (WAP)
  - bridge
- types of communication networks
  - personal area network (PAN)
  - local area network (LAN)
  - wide area network (WAN)
  - worldwide interoperability for microwave access (WiMAX)
  - wireless (PAN, LAN, WAN)

- technologies appropriate for the implementation of a client/server and peer-to-peer network
- star network topology
- diagrammatic representation of network topologies for LAN and WAN
- characteristics of transmission media, including:
  - twisted pair (unshielded twisted pair [UTP] and shielded twisted pair [STP])
  - fibre optic
  - satellite
  - microwave
  - cellular
  - wireless
- communication terms
  - protocols
  - digital
  - analogue
  - ethernet
  - bandwidth
- types of communication protocols, including:
  - file transfer protocol (FTP)
  - hypertext transfer protocol (HTTP)
  - hypertext transfer protocol secure (HTTPS)
  - simple mail transfer protocol (SMTP)
  - wireless access protocol (WAP)
- 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 LAN 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 11 syllabus and the weighting for each assessment type.

Assessment table – Year 11

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 stimulus material on which the project is based. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; flow charts; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and/or programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or Level 0 logical data flow 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 production.	40%
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; flow charts; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and/or programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or Level 0 logical data flow diagrams. 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, spreadsheets, 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; flow charts; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of spreadsheets, databases and/or programs. Diagrams can include: entity relationship diagrams, computer system diagrams, network diagrams, context diagrams and/or Level 0 logical data flow diagrams.	20%
Practical test Tests typically consist of a set of questions requiring the use of spreadsheet software, a programming language and/or a relational database management system. Spreadsheet skills assessed include creating spreadsheets that include functions, charts, lookup tables and sorting data. 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. In preparation for Unit 3 and Unit 4, the examination should reflect the examination design brief included in the ATAR Year 12 syllabus for this course.	30%

Teachers are required to use the assessment table to develop an assessment outline for the pair of units (or for a single unit where only one is being studied).

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. In the assessment outline where a single unit is being studied, each assessment type must be included at least once, 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 1 and Unit 2. At least one practical programming task and one practical database task must be conducted under test conditions across the year.

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 (or for a unit where only one unit is being studied). 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 11 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 11

A	**Knowledge and understanding** Accurately uses computer science terminology, and describes processes and concepts in context and with justifications.
	Systems development processes Compares systems development methodologies and justifies an appropriate method for developing a system. Conducts a detailed analysis of an information system using an appropriate methodology, uses system requirements to recommend and justify a digital solution and presents alternative solutions where relevant. Accurately represents information systems using context and Level 0 data flow diagrams.
	Data management skills Consistently constructs appropriate entity relationship diagrams accurately reflecting system requirements, including relevant use of diagrammatic conventions, relationships, cardinality, attributes and primary and foreign keys. Consistently constructs efficient and appropriate multi-table relational databases accurately reflecting system requirements and uses a range of appropriate query techniques to extract relevant data. Consistently constructs accurate spreadsheet solutions reflecting system requirements, using appropriate functions, charts, lookup tables and sorting.
	Programming skills Designs and creates flowcharts and pseudocode to represent complex algorithms with appropriate use of standards and conventions. Accurately applies appropriate programming control structures in pseudocode and a programming language accurately reflecting system requirements. Successfully designs, creates and applies appropriate test data and trace tables to accurately and efficiently validate data and resolve algorithmic logic. Consistently and successfully applies the software development cycle to create an effective software solution that accurately reflects system requirements.

B	**Knowledge and understanding** Accurately uses computer science terminology, and describes processes and concepts in context.
	Systems development processes Compares systems development methodologies and attempts to justify an appropriate method for developing a system. Conducts an analysis of an information system using an appropriate methodology, and recommends and justifies a digital solution based on relevant system requirements. Appropriately represents information systems using context and Level 0 data flow diagrams.
	Data management skills Constructs appropriate entity relationship diagrams, reflecting system requirements including relevant use of diagrammatic conventions, relationships, cardinality, attributes and primary and foreign keys. Constructs appropriate multi-table relational databases accurately reflecting system requirements and using appropriate queries to extract relevant data. Constructs functional spreadsheets solutions reflecting system requirements, using functions, charts lookup tables and sorting.
	Programming skills Designs and creates flowcharts and pseudocode with appropriate use of standards and conventions. Applies appropriate programming control structures in pseudocode and a programming language to reflect system requirements. Designs, creates and applies appropriate test data and trace tables to accurately validate data and resolve algorithmic logic. Successfully applies the software development cycle to create an appropriate software solution that reflects system requirements.

C	**Knowledge and understanding** Uses computer science terminology, and describes processes and concepts.
	Systems development processes Describes the types of systems development methodologies and applies a method to create a system. Conducts an analysis of an information system using an appropriate methodology, and recommends a digital solution based on limited system requirements. Represents information systems using context and Level 0 data flow diagrams with minimal logic and convention errors. Correctly interprets context and data flow diagrams.
	Data management skills Constructs entity relationship diagrams, reflecting system requirements, including use of diagrammatic conventions, relationships, cardinality, attributes and primary and foreign keys. Correctly interprets entity relationship diagrams. Constructs a multi-table relational database, reflecting system requirements and using simple queries to extract data. Constructs spreadsheet solutions, reflecting system requirements, using a limited range of functions, charts, lookup tables and/or sorting.
	Programming skills Designs and creates flowcharts and pseudocode with appropriate use of standards and conventions. Applies programming control structures in pseudocode and a programming language, including minor errors in logic, code and/or convention, reflecting a limited range of system requirements. Designs, creates and applies test data and trace tables to validate data and resolve algorithmic logic. Applies the software development cycle to create an appropriate software solution that reflects system requirements.

D	**Knowledge and understanding** Attempts to describe computer science terminology, processes and concepts.
	Systems development processes Correctly lists stages in the Systems Development Life Cycle. Develops planning and presentation which is brief or incomplete and indicates limited understanding of the information system or system requirements. Conducts analysis of an information system which is incomplete and recommendations that are brief. Attempts to represent information systems, using context diagrams with logic and convention errors. Attempts to interpret context diagrams.
	Data management skills Constructs inaccurate entity relationship diagrams with significant logic and convention errors. Inaccurately interprets entity relationship diagrams. Constructs an incomplete database with limited functionality. Constructs an incomplete spreadsheet with limited functionality.
	Programming skills Attempts to design and create flowcharts and represents incomplete algorithms using pseudocode with standard and convention errors. Develops algorithms and programs which are incomplete and inaccurate demonstrating little understanding of programming control structures. Creates trace tables which are incomplete, inaccurate or unable to test algorithmic logic. Creates test data which is incomplete, inaccurate or unable to test program functionality. With guidance, applies the software development cycle, partially creating a software solution that reflects a minimum set of 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.