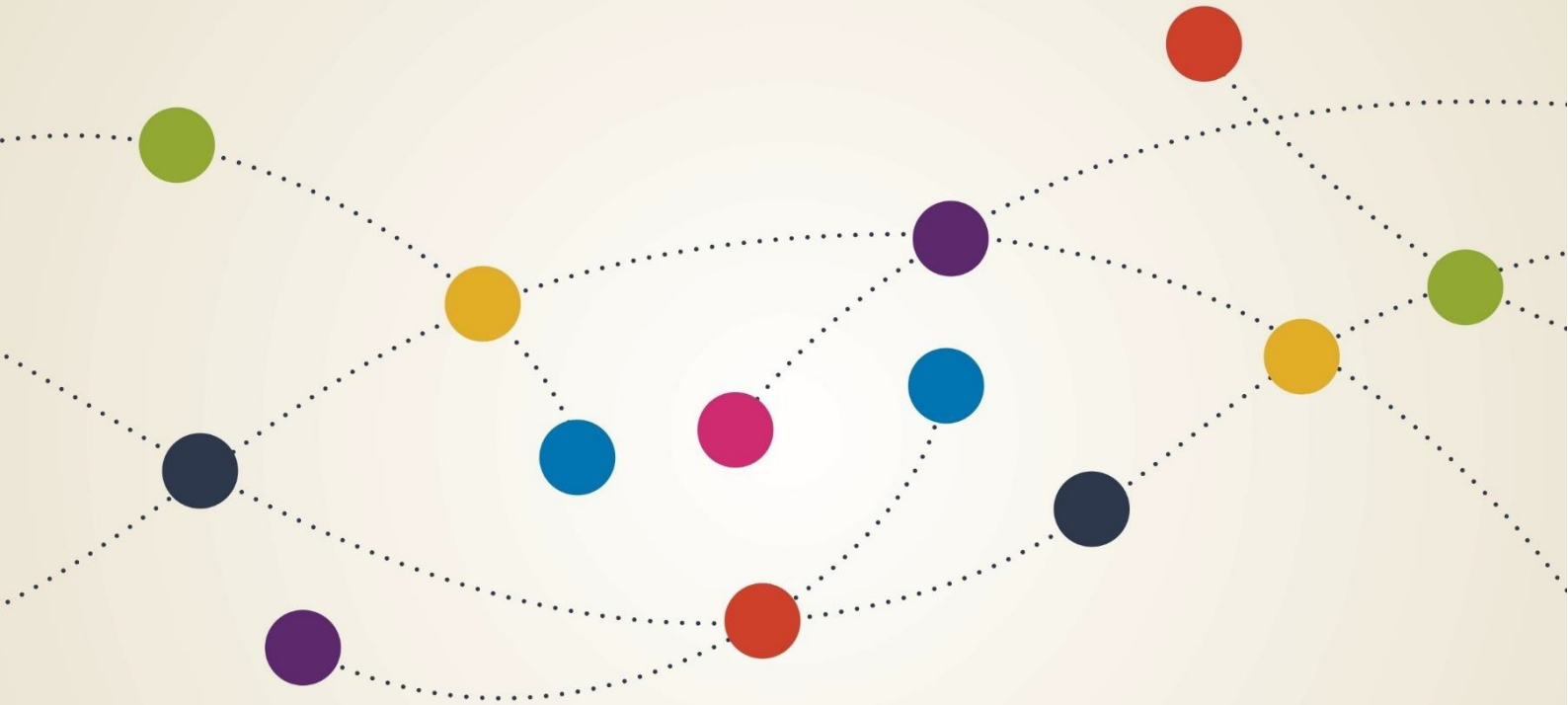


Leaving Certificate Computer Science Draft Specification

For consultation

NCCA
An Chomhairle Náisiúnta Curaclaim agus Measúnachta
National Council for Curriculum and Assessment

Contents

Senior cycle 4

The experience of senior cycle 4

Leaving Certificate Computer Science 8

Introduction 8

Aim 8

Objectives 8

Related learning 9

Early Childhood 9

Primary school 9

Junior cycle 10

Senior cycle 10

Further study 10

Society and community 11

Education for sustainable development 11

Structure 12

Strand 1: Practices and Principals of Computer Science 13

Strand 2: Cross-cutting core concepts 13

Strand 3: Computer Science in Practice 13

Key skills 14

Teaching and learning 16

Project work 16

Time allocation 18

Differentiation 18

Strands 20

Assessment 27

Assessment for certification 27

Coursework assessment 30
Leaving Certificate Grading 32
Reasonable Accommodations/Inclusion 33

Table of figures

Figure 1 Overview of senior cycle 6
Figure 2 The vision of the learner 7
Figure 3 Structure of Leaving Certificate Computer Science 12
Figure 4 Key Skills of Senior Cycle 14
Figure 5 The design process 17

Senior cycle

Students in senior cycle are approaching the end of their time in school and are focusing on the directions they would like to take in their future lives. Senior cycle plays a vital role in helping students to address their current needs as young adults and in preparing them for life in a changing economic and social context.

Senior cycle is founded on a commitment to educational achievement of the highest standard for all students, commensurate with their individual abilities. To support students as they shape their own future there is an emphasis on the development of knowledge and deep understanding; on students taking responsibility for their own learning; on the acquisition of key skills; and on the processes of learning. The broad curriculum, with some opportunities for specialisation, supports continuity from junior cycle and sets out to meet the needs of students, some of whom have special educational needs, but who all share a wide range of learning interests, aptitudes and talents.

Curriculum components at senior cycle promote a balance between knowledge and skills, and the kinds of learning strategies relevant to participation in, and contribution to, a changing world where the future is uncertain.

Assessment in senior cycle involves gathering, interpreting and using information about the processes and outcomes of learning. It takes different forms and is used for a variety of purposes. It is used to determine the appropriate route for students through a differentiated curriculum, to identify specific areas of difficulty or strength for a given student and to test and certify achievement. Assessment supports and improves learning by helping students and teachers to identify next steps in the teaching and learning process.

The experience of senior cycle

The vision of senior cycle sees the learner at the centre of the educational experience. That experience will enable students to be resourceful, to be confident, to participate actively in society, to build an interest in learning, and develop an ability to learn throughout their lives.

This vision of the learner is underpinned by the values on which senior cycle is based and it is realised through the principles that inform the curriculum as it is experienced by students in schools. The curriculum, made up of subjects and courses, embedded key skills, clearly expressed learning outcomes, and supported by a range of approaches to assessment, is the vehicle through which the vision becomes a reality for the learner.

At a practical level, the provision of a high quality educational experience in senior cycle is supported by:

- effective curriculum planning, development, organisation and evaluation
- teaching and learning approaches that motivate and interest students, that enable them to progress, that deepen and apply their learning, and that develop their capacity to reflect on their learning
- professional development for teachers and school management that enables them to lead curriculum development and change in their schools
- a school culture that respects students, that encourages them to take responsibility for their own learning over time, and that promotes a love of learning.

Senior cycle education is situated in the context of a broader education policy that focuses on the contribution that education can make to the development of the learner as a person and as a citizen. It is an education policy that emphasises the promotion of social cohesion, the growth of society and the economy, and the principle of sustainability in all aspects of development.

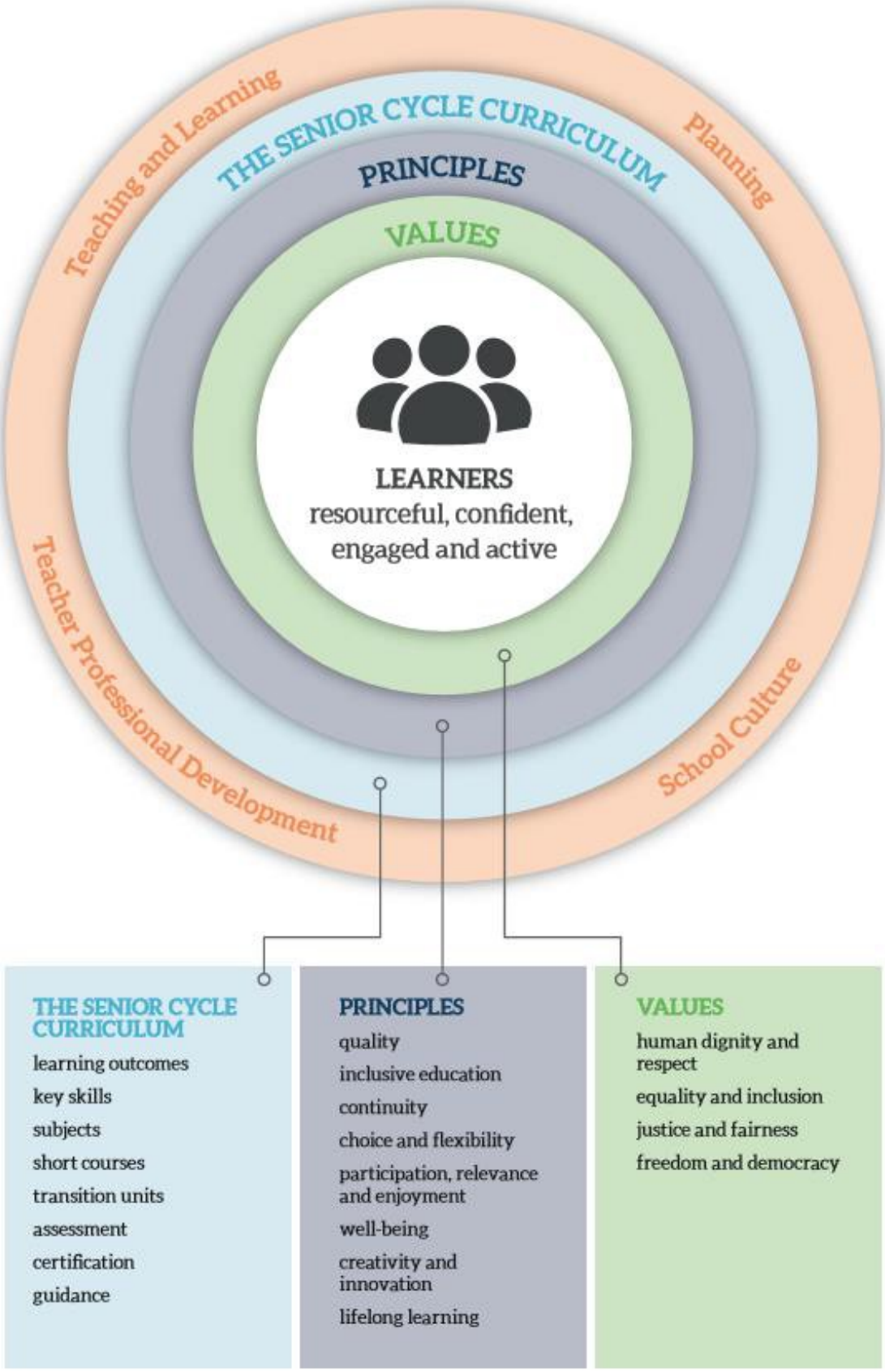


Figure 1 Overview of senior cycle

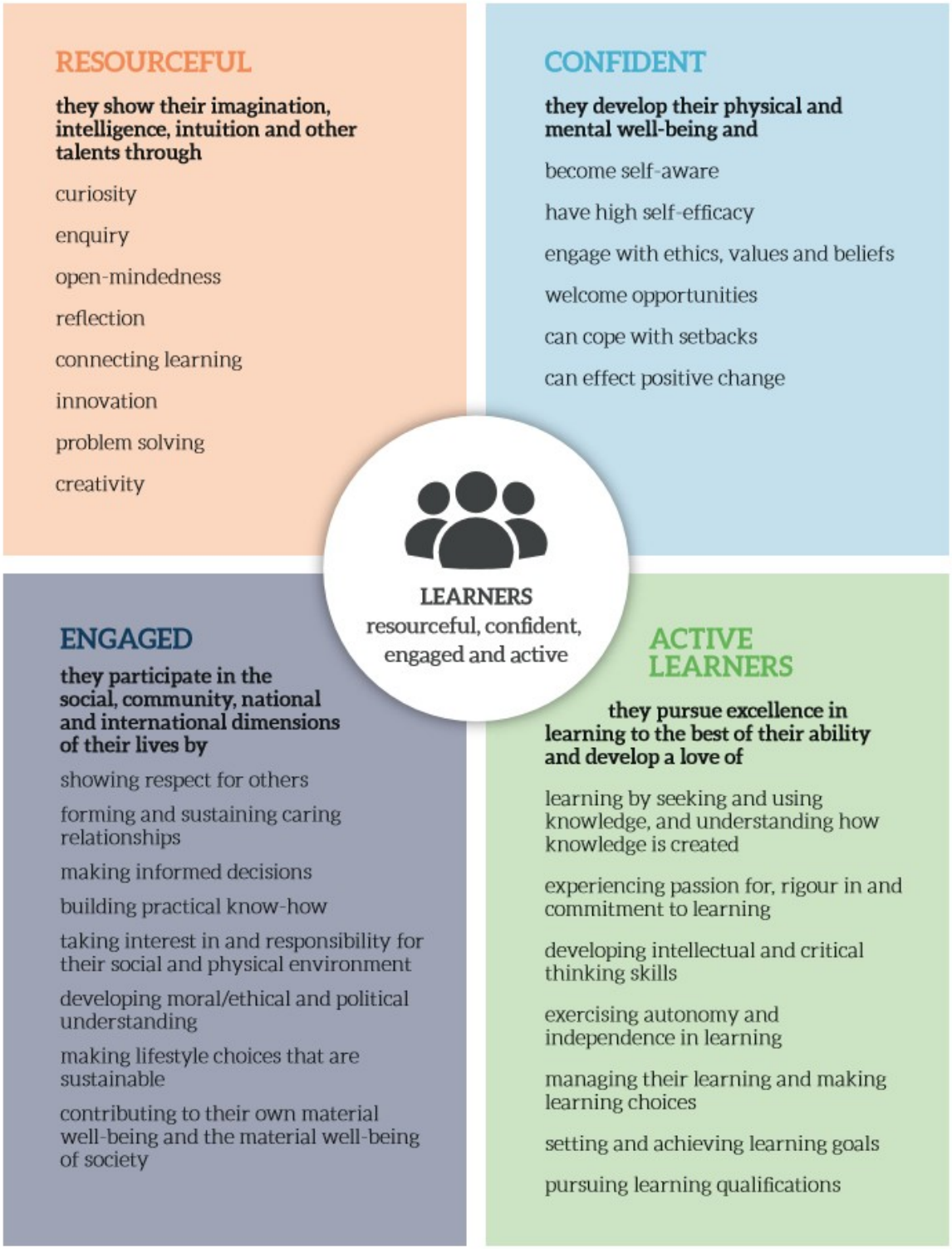


Figure 2 The vision of the learner

Leaving Certificate Computer Science

Introduction

Computer science is the study of computers and algorithmic processes. It includes the principles and practices of computer science, as well as hardware and software designs, computer applications, and their impact on society. Computational thinking is the underlying problem-solving process that drives computer science and connects it to every other discipline. Programming (writing code) is the form in which computing instructions are written. As they study computer science, students will develop their understanding of cross-cutting core concepts such as abstraction, data structures, and algorithms. They will enhance their problem-solving capabilities through the application of computational thinking, and learn to develop computational artefacts and apply effective time management and software development skills. These practices and skills will enable students to understand the relevance of computers and their potential to enhance society and the environment in which we live.

Computer science students learn logical reasoning, algorithmic thinking, design and structured problem solving — all concepts and skills that are valuable well beyond the computer science classroom and are applicable in many contexts, from science and engineering to the humanities and business. As they study computer science, students will use a wide range of available tools, as well as the principles of computer science to develop appropriate and ethical solutions to problems, and communicate these solutions in a coherent persuasive manner. Computer science also helps us better understand phenomena in the world around us and our relationship to it.

The computer science specification is designed for all students; it applies to virtually every aspect of students' lives; therefore, it is explicitly relevant to the vast array of student interests.

Aim

Leaving Certificate Computer Science aims to develop and foster the learner's creativity and problem solving, along with their ability to work both independently and collaboratively. Students will apply the fundamental practices and concepts of computer science and develop an appreciation of the diverse role of computing technology in society and the environment in which they live. Studying computer science will nurture students' interests and passions and empower them to engage confidently and actively with the world.

Objectives

The objectives of Leaving Certificate Computer Science are to enable students to:

- develop an understanding of how computing technology presents new ways to address problems; and to use computational thinking to analyse problems; and to design, develop and evaluate solutions
- read, write, test, and modify computer programs

- develop an understanding of how computers work; the component parts of computer systems and how they interrelate, including software, data, hardware, communications, and users
- appreciate the ethical and social implications relating to use of computing technology and information and identify the impact of technology on personal life and society
- understand how information technology has changed over time and the effects these changes may have on education, the workforce, and society
- evaluate the accuracy, relevance, appropriateness, comprehensiveness, and bias of online information sources
- work independently and collaboratively, communicate effectively, and become responsible, competent, confident, reflective, and creative users of computing technology.

Related learning

Leaving Certificate Computer Science builds on the knowledge, attitudes and broad range of transferable skills that stem from the student's educational experience at early childhood, primary and post-primary junior cycle levels.

Early Childhood

Aistear, the early childhood curriculum framework, celebrates early childhood as a time of well-being and enjoyment where children learn from experiences as they unfold. The theme of Exploring and Thinking is about children making sense of the things, places and people in their world by interacting with others, playing, investigating, questioning, and forming, testing and refining ideas. The theme of Communicating is about children sharing their experiences, thoughts, ideas, and feelings with others with growing confidence and competence in a variety of ways and for a variety of purposes.

Primary school

The integrated programme of Social, Environmental and Scientific Education (SESE) at Primary school provides opportunities for children to actively explore and investigate the world around them from a human, social and cultural perspective. A scientific approach to investigations fosters the development of important skills, concepts and knowledge through which children can observe, question, investigate, understand and think logically about living things and their environments, materials, forces, everyday events and problems. The knowledge and skills acquired may be applied in designing and making activities in which children perceive a need to create or modify elements of their environments. Through their investigations, children develop informed, critical and scientific perspectives that acknowledge the importance of founding judgements on a respect for facts, accuracy and reason.

Computer science builds on language skills developed at primary level. Through language, students learn to use appropriate sequencing, tense, and vocabulary to tell and retell stories and personal and procedural narratives of increasing complexity. They learn to use topic-specific language to give information, to explain and to justify their ideas and to predict and reflect upon actions, events and processes relating to real and imaginary contexts. Language skills, developed at primary level will help students of computer science to appreciate the importance of the correct use of language, and appreciate the value of how powerful words and language are in the context of social media.

The Mathematics curriculum at Primary school aims to provide children with a language and a system through which to analyse, describe, illustrate, model and explain a wide range of experiences, make predictions, and solve problems. Leaving Certificate Computer science builds on these skills, as it supports students to think and communicate quantitatively and spatially, solve problems, recognise situations where mathematics can be applied.

Junior cycle

Leaving Certificate Computer Science builds on, and enhances the Key Skills of Junior Cycle :

- Being Literate
- Managing Myself
- Staying Well
- Managing Information and Thinking
- Being numerate
- Being creative
- Working with others
- Communicating

Senior cycle

Many senior cycle subjects have close links with Computer Science. Computational Thinking is a thought process (or a human thinking skill) that uses analytic and algorithmic approaches to formulate, analyse and solve problems. Whilst the alignment of computer science with the STEM subjects is obvious, the strategies learned in Computer Science also relate to learning in other subjects. For example, Computer Science shares similarities with language learning as pattern recognition, syntax, textual analysis and argument formation are relevant to both fields of study. Computer Science provides a context for students to develop metacognitive skills which will support students as they take responsibility for their own learning.

Further study

Students live in a technologically rich world. Leaving Certificate Computer Science will provide students with the knowledge and skills that will help them to understand current computer technology and prepare them for emerging technologies. A foundation in this discipline will introduce students to the excitement and opportunities afforded by this growing and dynamic field, as well as preparing them for a range of rewarding careers.

Leaving Certificate Computer Science incorporates a broad range of transferable and trans-disciplinary skills such as: problem- solving, logical thinking, and creative design. It also promotes skills of synthesis, evaluation, communication, time management, organisation, and teamwork. These skills and capabilities provide support for further study and learning beyond formal education. Including: computer programming, database analysis, computer science, computer engineering, software engineering, information technology, and game development.

Society and community

Leaving Certificate Computer Science includes the study and discussion of current events and emerging technologies, which will stimulate student interest and curiosity, and help them connect what they are learning in class with real-world events or situations. Exploring the benefits and drawbacks of current and future computing technologies, and most importantly their impact on people and societies, will help students develop and refine their understanding of how to use computing technology and information ethically. Additionally, students will explore the role that adaptive technology can play in the lives of people with special needs and of how access to and engagement with the world of, and outcomes of, computing is of ever-increasing importance to societies, democracies and human progress.

Community links are an important resource for schools and students participating in Leaving Certificate Computer Science. These links can take the form of industry-local business mentoring/career programmes, university mentoring programmes, participating in/leading local code clubs or school coding clubs, and collaborating with local community groups to use technology to solve a local problem.

Education for sustainable development

The *National Strategy on Education for Sustainable Development 2014 -2020* highlights the need to integrate Education for Sustainable Development (ESD) in the curriculum from pre-school up to senior cycle. The National Strategy aims to ensure that education contributes to sustainable development by equipping learners with the relevant knowledge (the 'what'), the key dispositions and skills (the 'how') and the values (the 'why') that will motivate and empower them throughout their lives to become informed active citizens who act for a more sustainable future.

This Computer Science specification supports education for sustainable development by integrating the key skills of senior cycle throughout the strands. Many of the contexts used to explore the knowledge and understanding of computer science provide opportunities to discuss the practical and ethical aspects of computing, and to consider the use of computers and related technology from a societal perspective.

In strand 1, the practices and principles of computer science are encountered in a context-based approach related to social, professional, and scientific contexts. Students will appreciate how the use of computing technology impacts communities. In strand 2. students learn how solutions can be designed that exploit the power of computers; they will consider ethical dilemmas and contexts relating to the use of computers, including how the resources used in the product life cycle –water, fuel, and electricity can be compensated for by their ability to increase energy efficiency

by changing systems and ways of working. In strand 3, as students build computational artefacts, they appreciate the possibilities of how computing technology can provide ways to protect natural resources. For example, how modelling can be used to optimise systems to improve efficiency and reduce the damaging impact of energy consuming infrastructures and systems. Throughout the course, students will apply the fundamental practices and concepts of computer science and develop an appreciation of the diverse role of computing technology in society and the environment in which they live.

Structure

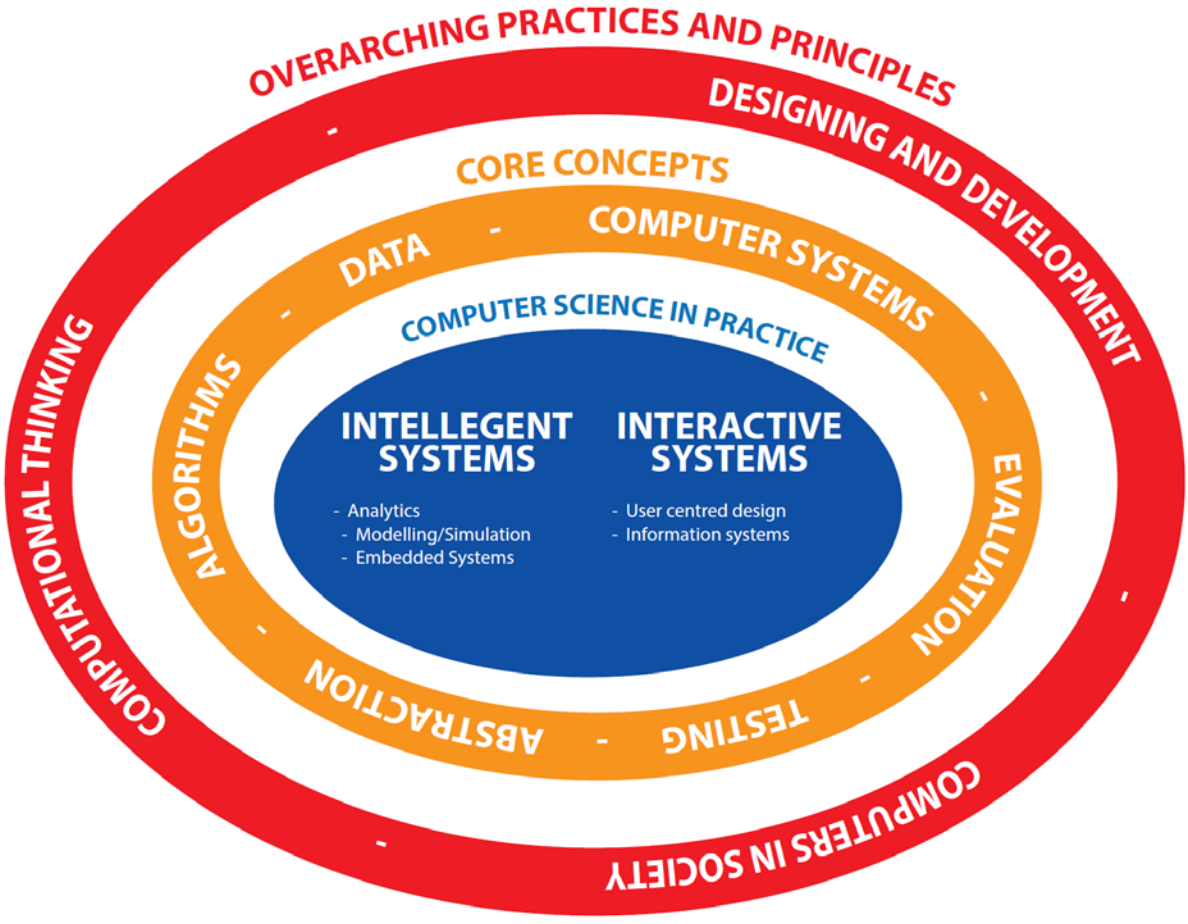


Figure 3 Structure of Leaving Certificate Computer Science

The three strands of the specification: *Practices and Principles, Cross-cutting Core Concepts and Computer Science in Practice* are fundamentally interrelated. Throughout the specification, students engage in the fundamental *practices and principles* of computer science as they apply the *crosscutting core concepts* to real life contexts. In Strand 3, *computer science in practice,* students develop computational artefacts which provide the contexts for them to work within the parameters of the practices and principles of Computer Science to develop conceptual, procedural and metacognitive understanding.

Strand 1: Practices and Principals of Computer Science

The overarching *Practices and principles* of computer science are the behaviours and ways of thinking that computer scientists use. This strand underpins the specification, and is fundamental to all learning activities. By becoming familiar with, and fluent in, the practices and principles that underpin good practice, students develop their ability to manage themselves and their learning.

Strand 2: Cross-cutting core concepts

The Cross-cutting core concepts of computer science represent major content areas in the field of computer science: abstraction, data, computer systems, algorithms and evaluation/testing. Students engage with the cross-cutting concepts theoretically in this strand and apply them practically in Strand 3, Computer Science in Practice. In this way, conceptual classroom-based learning is combined with experimental computer-based learning throughout the two years of the course.

Strand 3: Computer Science in Practice

Computer science in practice provides multiple opportunities for students to apply the practices and principles and the cross-cutting core concepts. Students work in groups to carry out five projects, each of which results in the creation of specified computational artefacts1. These artefacts are personally relevant or beneficial to the community and society in general. Examples of computational artefacts include programs, simulations, visualisations, digital animations, robotic systems, and apps.

The five projects are grouped under two headings: Interactive systems and Intelligent systems. The projects are sequenced in such a way that the first project is at an introductory level, and subsequent projects provide opportunities for students to develop their theoretical and procedural understanding as they grapple with Computer Science practices and principles and the Cross-cutting core concepts in increasingly sophisticated applications. The output from each project is a specified computational artefact and a report outlining its development. In the report, students outline where and how the core cross-cutting concepts were employed. The reports are collected in a digital portfolio along with the computational artefact they describe. A description of the content of the report is described on pages 16 – 17. Sample projects will be available on www.curriculumonline.ie

[1] A computational artefact is anything created by a human using a computer. An artefact can be, but is not limited to, a program, image, audio, video, presentation, or web page file.

Key skills

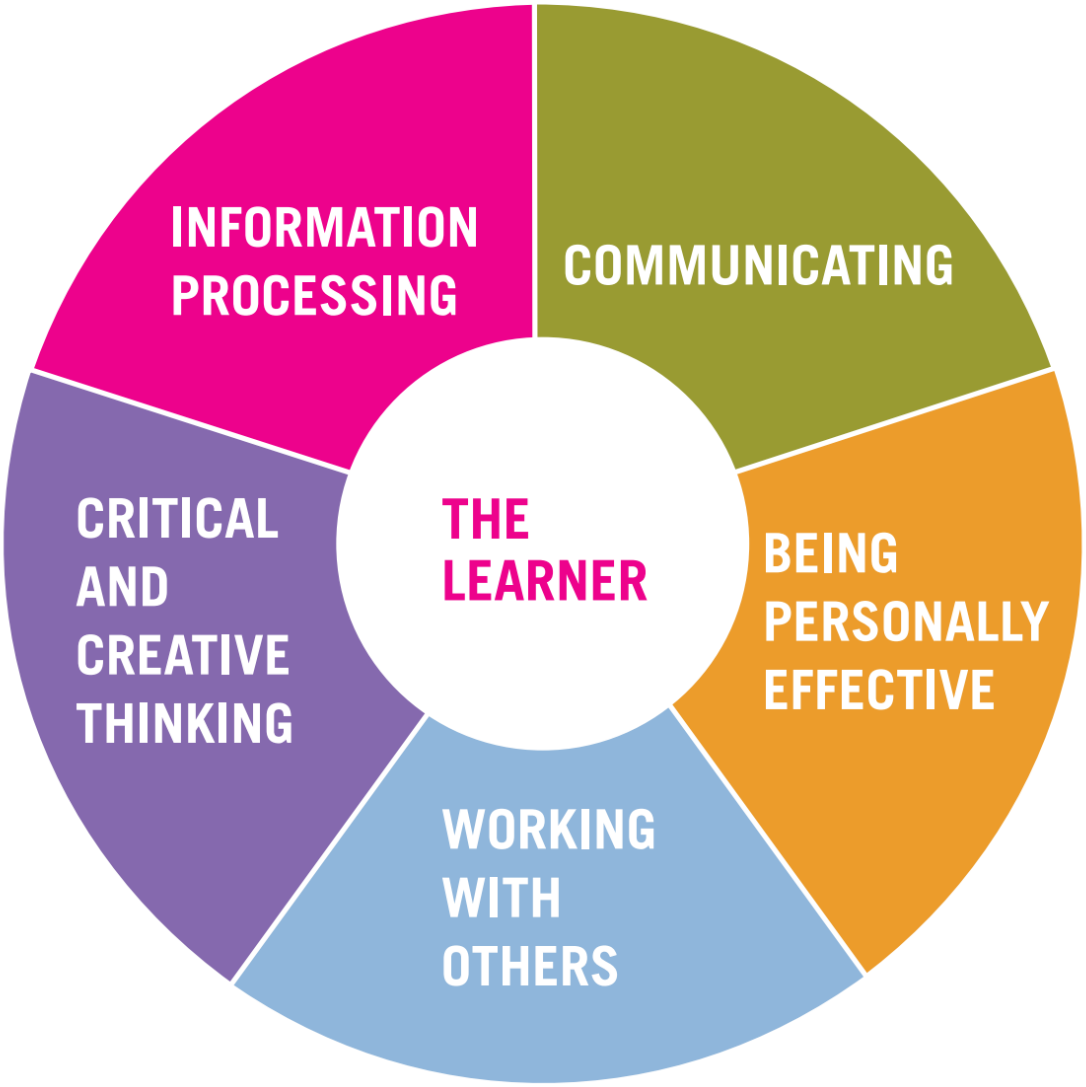


Figure 4 Key Skills of Senior Cycle

Recent developments in curriculum and assessment at senior cycle have focused on the embedding of key skills within learning outcomes. This is accompanied by a different approach to assessment in which students can generate responses that reveal the depth of their understanding. The embedding of key skills requires careful consideration of the balance between knowledge and skills in the curriculum and in learning and of finding appropriate ways of assessing them. The key skills of: *information processing; being personally effective; communicating; critical and creative thinking* and *working with others*, and the learning outcomes associated with them, comprise the NCCA Key Skills Framework (NCCA, 2009). The Key Skills Framework was developed to provide a common, unified approach for embedding key skills across all future Leaving Certificate specifications. From an Irish perspective, these skills were identified as being important for all students to achieve to the best of their ability, both during their time in school and into the future, and to fully participate in society, in family and community life, the world of work and lifelong learning. Computer Science develops these skills in the following ways:

Information processing

Learning in computer science takes place in an information-intensive environment; it promotes independent research activities in which students are required to access a wide variety of external materials to tackle questions. Tasks in computer science address selection, evaluation and recording of information; as students engage in problem-solving, they make decisions and judgments based on data and qualitative and quantitative information. In this information-intensive environment, students develop an appreciation of the differences between information and knowledge and the roles that both play in making decisions and judgements. Programming teaches respect for accuracy and attention to detail. The consequence of a lack of precision is that the program fails or produces inaccurate or inconsistent results for different inputs. Similarly, the idea of breaking down a problem into sub-problems that can be solved separately takes a very concrete form in Computer Science, in which sub-systems interact through carefully-defined interfaces.

Critical and creative thinking

Modelling, programming, and coding require careful analysis of patterns and relationships which develops skills of higher-order reasoning and problem solving. Part of computational thinking is the ability to identify, analyse and deconstruct problems, explore options and alternatives, and hence solve problems. Hypothesising, making predictions, examining evidence, and reaching conclusions underpins the core of all the activities that students will undertake as part of a computer science course. As they develop these skills, students reflect critically on the forms of thinking and values that shape their own perceptions, opinions and knowledge. This develops the metacognitive dimension of knowledge which is essential in promoting good habits of mind. In computer science, students are designers and creators of technology rather than merely users of technology. Students can be creative and expressive through the creation of artefacts.

Communicating

Strong communication skills are developed in collaborative project work. Students use technology to communicate face to face and through digital media. Although literacy skills are not targeted directly, they are required to participate fully in the learning experience. Internet research and use of external models require and builds, analysis and interpretation skills. Students will read a wide range of information sources. As part of the course students will be required to express and share their opinions through debate and argument. This encourages engaging in dialogue, listening attentively and eliciting opinions, views and emotions. They will also learn to provide technical information in a way that is relevant to and understandable by people with diverse levels of technical knowledge and understanding. There is opportunity to develop communication skills further as students compose and present using a variety of media.

Working with others

Leaving Certificate Computer Science is underpinned by collaboration and working with others. In their project work, students gain appreciation of the dynamics of groups and the social skills needed to engage in collaborative work. Computer Science contributes to an appreciation that working collectively can help motivation, release energy and capitalise on all the talents in a group. One of the crucial factors in working with others is to identify, evaluate and

achieve collective goals. Students learn to negotiate and resolve conflicts as they discuss their different strategies and achieve consensus.

Being personally effective

Self-awareness and persistence in the face of challenges enable students to grow and to develop. In computer science, students work on uncertain problems, and learn to persist through ambiguity and face the risk of failure. As they work though challenges and potential failure they build persistence and resilience which serves them in all areas of their lives. There is no right way to answer a problem or set up a problem-solving strategy; as students develop confidence in their self-direction they develop tenacity and rigour. By developing the skill of being personally effective, students develop strategies to learn and to build on previous knowledge.

Teaching and learning

Senior cycle students are encouraged to develop the knowledge, skills, attitudes, and values that will enable them to become independent learners and to develop a lifelong commitment to improving their learning. Leaving Certificate Computer Science supports the use of a wide range of teaching and learning approaches. As students progress they will develop learning strategies that are transferable across different tasks and different subjects enabling them to make connections between computer science, other subjects, and everyday experiences. Through engaging in self-directed learning activities and reflection students will plan, monitor, and evaluate their own learning and develop a positive sense of their own capacity to learn. By engaging in group work students will develop skills in reasoned argument, listening to each other, informing one another of what they are doing, and reflecting on their own work and that of others. They will develop skills in communication by collaborating to generate reports and present them to their peers. The projects will enable students to take an active role in their own learning by setting goals, developing action plans, and receiving and responding to assessment feedback.

Project work

Computer Science is centred around computing concepts that support the creation of exciting and relevant computational artefacts. The projects that students undertake during the two years of the course provide multiple and engaging opportunities for students to work within the practices and principles of computer science and apply the core cross-cutting concepts in practical applications. The five projects are grouped under two headings and are listed in order of complexity. The computational artefacts that students create are personally relevant to them, their community or to society in general. Examples of computational artefacts include programs, simulations, visualizations, digital animations, robotic systems, and apps. Over the course of the two years of computer science students will:

1. Create an artefact that meets specific user/ community needs
2. Create a website that can display information from a database - remote or local
3. Create an interdisciplinary artefact

4. Develop a computer system that simulates or models a problem that is difficult to solve analytically

5. Implement a microprocessor system that uses sensors and controls digital inputs and outputs within an embedded system

Project management

Project management is a core feature of the specification. As students complete their projects they will learn how to adopt distinct roles and adapt to changing situations. Students assume different responsibilities in each project, rotating between the roles of team leader, project manager, communications manager, programmer.

The design process

Students collaborate to manage their projects using the design process illustrated in figure 4.

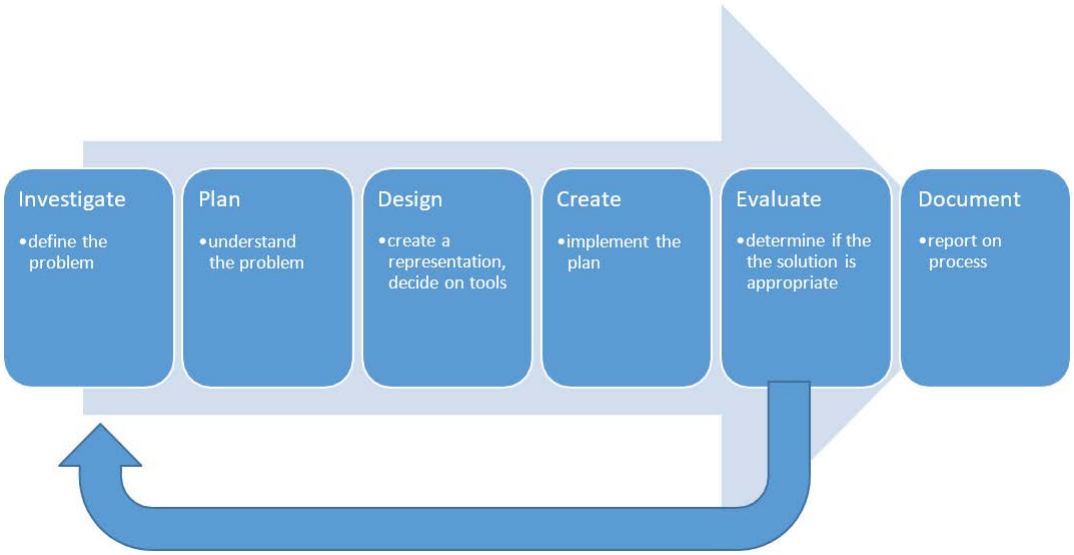


Figure 5 The design process

The process is iterative and cyclical and students are encouraged to adapt their design in light of evaluation and testing at each stage of the process. The core concepts (indicated by italics below) are encountered through each of the five projects. In the investigation phase, students use analysis to generate questions and computational solutions. They seek relevant information in articles, books and other sources, as well as by examining data. In the planning phase students use abstraction to identify tasks and select appropriate strategies to create the artefact. In the design phase, students create visual representations of a model and decide which tools to use and which algorithms to use, adapt or create as they use appropriate techniques to develop their solution. In the create phase, students develop *computer systems* as they use programming, analysis and design skills combined with hardware knowledge to create network/Internet/cloud based applications. In the *evaluate* phase, students identify and remove errors from their programmes and check that their artefact does what it is supposed to do. They learn from their mistakes to effectively solve new problems in different situations. Once the artefact is created, students report on the process of the creation as well as on the artefact itself.

Time allocation

Computer Science is designed for 180 hours of class contact time. There is time for introductory classroom-based theory, and revision, but most of the teaching and learning will take place in the practical application of the concepts in strand 3. Each project should take approximately six weeks.

Differentiation

The Leaving Certificate Computer Science specification is differentiated in three ways: through the learning outcomes of the specification, in the process of teaching and learning, and through assessment.

Differentiation through the learning outcomes of the specification

Ordinary Level	Higher Level
Only the learning outcomes that are presented in normal type Students engage with a broad range of knowledge, mainly concrete in nature, but with some elements of abstraction or theory. They will be expected to demonstrate and use a moderate range of practical and cognitive skills and tools and to plan and develop simple investigative strategies. They will be expected to select from a range of procedures, and apply known solutions to a variety of problems in both familiar and unfamiliar contexts. They will design and produce computational artefacts that serve a useful purpose.	All learning outcomes including those in **bold type** Students engage with a broad range of knowledge including theoretical concepts and abstract thinking with significant depth in some areas. They will be expected to demonstrate and use a broad range of specialised skills and tools to evaluate and use information, to plan and develop investigative strategies, and to determine solutions to varied, unfamiliar problems. They will be expected to identify and apply skills and knowledge in a wide variety of both familiar and unfamiliar contexts. They will design and produce computational artefacts that serve a useful purpose.

Table 2 Differentiation in Leaving Certificate Computer Science

Differentiation in teaching and learning

Students vary in the amount and type of support they need to be successful. Levels of demand in any learning activity will differ as students bring different ideas and levels of understanding to it. The use of strategies for differentiated learning such as adjusting the level of skills required, varying the amount and the nature of teacher intervention, and varying the pace and sequence of learning will allow students to interact at their own level.

Differentiation in assessment

Assessment of Leaving Certificate Computer Science will be based on the learning outcomes in the specification. The computer-based, end-of-course examination will be assessed at two levels, Higher and Ordinary. At Higher Level, all the learning outcomes will be assessed including those presented in bold type. At Ordinary Level, only those learning outcomes that are presented in normal type will be assessed. Examination questions will require candidates to demonstrate knowledge, understanding, application, analysis, and evaluation appropriate to each level. Differentiation at the point of assessment will also be achieved through the language register of the questions asked, the stimulus material used, and the extent of the structured support provided for examination candidates at different levels.

Strands

Strand 1: Practices and Principles

The practices and principles of computer science describe the behaviours and ways of thinking that computationally literate students use to fully engage in a data-rich and interconnected world. Computational thinking, at the heart of computer science practices, is a problem-solving process that involves designing solutions that exploit the power of computers. The practices and principles are encountered in a context-based approach related to social, professional, and scientific contexts. Studying the role of computers in society will enhance students' attitudes towards computer science and make it more meaningful and relevant. In learning about designing and developing, students will recognise the creative challenge involved in creating artefacts and project management.

Students learn about	**Students should be able to:**
Computational Thinking	1.1. describe a systematic process for solving problems and making decisions
	1.2. explain how the power of computing enables different solutions to difficult problems
Problem solving	1.3. solve problems by deconstructing them into smaller units using a systematic approach in an iterative fashion
Logical thinking	1.4. solve problems using skills of logic
	1.5. evaluate alternative solutions to computational problems
Algorithmic thinking	1.6. explain the operation of a variety of algorithms
	1.7. develop algorithms to implement chosen solutions
	1.8. evaluate the costs and benefits of the use of computing technology in automating processes
	1.9. use modelling and simulation in relevant situations
	1.10. discuss when heuristics should and could be used and explain the limitations of using heuristics

Computers and society Social and ethical considerations of computing technologies	1.11. discuss the complex relationship between computing technologies and society including issues of ethics 1.12. identify the past and present stages, and consider emerging trends in the evolution of computing technologies 1.13. consider the quality of the user experience when interacting with computers 1.14. compare the positive and negative impacts of computing on culture and society 1.15. describe the role that adaptive technology can play in the lives of people with special needs 1.16. appreciate the importance of designing inclusive and accessible computing technologies 1.17. recognise the diverse roles and careers that use computing technologies
Designing and developing Design process Working in a team, assigning roles and responsibilities Communication and reporting Software development and management	1.18. recognise the role of the design process in Computer Science 1.19. compare the features of staged and iterative design and development processes 1.20. collaborate and assign roles and responsibilities within a team to tackle a large computing project 1.21. communicate effectively with stakeholders about requirements and design decisions 1.22. use a range of methods for identifying patterns in information and ideas 1.23. identify and take account of alternative perspectives considering different disciplines, stakeholders, and end users 1.24. read, write, test, and modify computer programs 1.25. reflect on the design and development process

Strand 2: Cross-cutting Core Concepts

This strand introduces five cross-cutting core concepts that represent major content areas in the field of computer science: *abstraction, data, computer systems, algorithms and evaluation/testing.* The core concepts are developed theoretically and applied practically over five projects in Strand 3. In this way, conceptual classroom based learning is intertwined with experimental computer lab based learning throughout the two years of the course.

Students learn about[1]	**Students should be able to:**
Abstraction	2.1. use abstraction to describe systems and to explain the relationship between wholes and parts 2.2. identify recurring patterns that share abstract commonalities 2.3. demonstrate modular design to develop hardware or software modules that perform a specific function 2.4. illustrate examples of abstract models
Computer systems **CPU: ALU, Registers, Program counter, Memory** Operating System layers: Hardware, OS, Application, User **Turing Machines and Automata**	2.5. describe the different types of logic gates and explain how they can be arranged into larger units to perform more complex tasks **2.6.** describe the rationale for using binary in digital computing **and how to convert to decimal and vice versa** 2.7. describe the different components within a computer and the function of those components
Data 8-bit ASCII **Non-Roman character sets** Unicode: UTF-8, Emojis	2.8. use data types that are common to procedural high-level languages: Boolean, integer, real, char, string, date 2.9. use ASCII and Unicode character sets to encode/decode a message and consider the importance of having such standards

[1] The column *students learn about* lists the specific systems, data, algorithms and testing that will be used in the end-of-course assessment.

	2.10. collect, store and sort both continuous and discrete data
Algorithms Sorting: Simple sort, Insert Sort, **Bubble Sort, Quicksort** Search: Linear search, Binary Search Algorithmic complexity: Big O notation Machine Learning/Artificial Intelligence Agent-Based Modelling	2.11. describe what an algorithm is and give an example 2.12. use pseudo code to outline the functionality of an algorithm 2.13. construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems 2.14. develop a procedure to fulfil a specific requirement 2.15. apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm 2.16. assemble **complex** algorithms from elements that already exist or can be developed separately, by using functions, procedures and modules 2.17. explain the common measures of algorithmic efficiency using any algorithms they have studied 2.18. explain when and what machine learning and Artificial Intelligence algorithms might be used in certain contexts 2.19. explain the benefits of using agent-based modelling and how it can be used to demonstrate emergent behaviours
Evaluation/Testing Debugging Testing: **Unit test,** Function test, System test.	2.20. test outcomes of solutions and decisions both in the short and long term 2.21. Identify and fix/debug warnings and errors in computer code and modify as required **2.22.** Critically reflect and identify limitations in completed code **and suggest possible improvements** 2.23. explain the different stages in software testing

Strand 3: Computer Science in Practice

Computer science in practice provides multiple and continual opportunities for students to use their conceptual understanding in practical applications. There are five projects in total, grouped under two headings. The projects are listed in order of complexity. As part of the projects, students plan and develop computational artefacts that are personally relevant or beneficial to their community and society in general. Examples of computational artefacts include programs, simulations, visualizations, digital animations, robotic systems, and apps.

Project 1. Interactive systems – User-Centred Design

Design is one of the key practices and principles of Computer Science. As designers and creators of technology, students can be innovative and expressive through the creation of artefacts. In this project, students will design and create an interactive artefact that meets specific user needs. For example, this artefact could be a game or a web page or an interactive form. By identifying the needs and perspectives of different kinds of users, students will deepen their understanding of the design and development process.

Students learn about	**Students should be able to:**
User-Centred Design Abstraction User Interfaces and usability Computers and Society Design and Development Project management	3.1. understand and list user needs/requirements before defining a solution 3.2. list principles of good interface design 3.3. describe the role of a user interface and factors that contribute to its usability. **Discuss different factors of a user interface by comparing related interface**s 3.4. use an appropriate programming language to create an interactive artefact that meets user needs

Project 2. Interactive systems – Information Systems	
Computer Science is an information-intensive discipline that involves the selection, evaluation, recording and presentation of information. In this project, students will develop a website that can display information (either local or remote data) from a database. Through planning and designing an infrastructure that can display data, students will develop their knowledge of the role computing systems can play in communicating information with the world around them.	
Information systems Web infrastructure - Computer Network Protocols: HTTP, **TCP/IP, VOIP** Web design File systems and **relational** databases	3.5. explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components and communication protocols 3.6. create a basic **relational** database to store and retrieve a variety of forms of data types 3.7. develop a website that can display information from a database
Project 3. Intelligent systems – Analytics	
Hypothesising, making predictions, examining evidence, and reaching conclusions are at the heart of Computer Science. In this project, students will identify an interdisciplinary topic, develop a hypothesis and utilise existing resources to highlight the salient information and inform future decisions. By identifying, analysing, and deconstructing a problem, students will deepen their understanding of core practices and principles of Computer Science.	
Students learn about	**Students should be able to:**
Analytics Abstraction Data collection and analysis Interpret data Algorithms	3.8. collect, store and sort both continuous and discrete data and develop algorithms that can find the frequency, mean, median and mode of that data 3.9. structure and transform **raw** data to prepare it for analysis 3.10. represent data to effectively communicate in a graphical form 3.11. use algorithms when analysing and interpreting data to highlight the salient information and inform future decisions

Project 4. Intelligent Systems – Modelling /Simulation	
Modelling, programming, and coding require careful analysis of patterns and relationships to solve problems. In this project, students will engage with a problem that is difficult to solve analytically but is amenable to a solution using simulation or modelling. Students will develop a computer system that simulates or models the problem. Engaging with a problem in this way will heighten students' awareness and understanding of how algorithms can be used across a wide range of disciplines and subjects.	
Modelling/simulation Abstraction Algorithms	3.12. develop a model that will allow different scenarios to be tested 3.13. analyse and interpret the outcome of simulations before **and after modifications have been made**.
Project 5. Intelligent Systems – Embedded systems	
The design and application of computer hardware and software are a central part of Computer Science. In this project, students will implement a microprocessor system that uses sensors, and controls digital inputs and outputs within an embedded system. By building the component parts of a computer system, students will deepen their understanding of how computers work and how they can be embedded in our everyday environments.	
Students learn about	**Students should be able to:**
Embedded systems Computing inputs and outputs Basic electronics: voltage, **pull up/down resistors** Computer Systems	3.14. describe the difference between digital and analogue inputs 3.15. use and control digital inputs and outputs within an embedded system 3.16. measure and store data returned from an analogue input **3.17.** develop a program that utilises digital **and analogue** inputs 3.18. design automated applications using embedded systems

Assessment

As well as varied teaching strategies, varied assessment strategies will support learning and provide information that can be used as feedback so that teaching and learning activities can be modified in ways that best suit individual learners. By setting appropriate and engaging tasks, asking higher-order questions, giving feedback that promotes learner autonomy, assessment will support learning as well as summarising achievement.

There are several important aspects of computer science assessments to consider: the use of authentic tasks, the breadth of concepts being assessed, and the special role computers can play in delivering instruction and measuring performance. Compared to other subjects, computer science provides a unique opportunity to take advantage of online learning and computerised assessment. Students can create programs such as games, apps, and simulations within an environment that also collects data, analyses achievement, and communicates progress to both students and teachers.

Project-based/portfolio assessment of coursework can measure many of the computer science learning outcomes associated with performance. Coursework assessment provides students with opportunities to demonstrate their understanding in multiple ways that highlight their creativity, interests, and understanding.

Assessment for certification

Assessment for certification is based on the aim, objectives, and learning outcomes of this specification. Differentiation at the point of assessment is achieved through examinations at two levels – Ordinary level and Higher level.

There are two components to the assessment of Leaving Certificate Computer Science, an end-of- course computer-based examination and coursework. Both components reflect the relationship between the application of skills and the theoretical content of the specification.

The end-of-course assessment will comprise multiple choice, short answer, practical and extended response questions. The questions will assess both the core cross-cutting concepts and the computer practices and principles. The questions will be based on the learning outcomes in the specification; however, any question may address more than one learning outcome, or require students to combine information from across several areas of the specification.

The coursework assessment will require students to demonstrate proficiency in course content and skills that cannot be assessed by the end of course examination. The assessment will require students to create an innovative computational artefact, and to report on the work and process involved. Students must acknowledge (i.e. through citation, through attribution, by reference, and/or through acknowledgement in bibliographic entry) the source or author of all information or evidence taken from somebody else's work. Student work for both components will be submitted electronically and will be marked by the State Examinations Commission (SEC).

The assessment of both components will be aligned with the objectives of the specification, and assess the extent to which students:

- understand how computing technology presents new ways to address problems
- use computational thinking to analyse problems, and to design, develop and evaluate solutions
- can read, write, test and modify computer programs
- understand how computers work; the component parts of computer systems and how they interrelate, including software, data, hardware, communications, and users
- understand the evolution of computing technology and appreciate the ethical and social implications of the use of computing technology in contemporary and future social issues to enable them to make informed decisions
- work independently and collaboratively, communicate effectively, and become responsible, competent, confident, reflective and creative users of computing technology.

Structure of assessment for certification

There are two assessment components at each level, an end-of-course examination (70%) and coursework (30%).

Component	Percentage
End-of-course examination	**70**
A. short-answer questions, B. practical question C. structured questions	**20** **30** **20**
Coursework assessment	**30**
▪ Computational artefact with report	**30**
Total	**100**

Table 3 Overview of assessment

End-of-course examination

The computer-based, end-of-course examination will be made up of a range and balance of question types. The questions will require students to demonstrate knowledge, understanding, application, analysis, evaluation and creation appropriate to each level. The key skills are embedded in the learning outcomes and will be assessed in the context of the learning outcomes. The examination will assess:

- knowledge and recall of facts, principles and methods relating to computer science
- application of knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation and how to analyse problems in computational terms
- ability to write code and to compile, test and debug program code
- ability to evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions
- problem solving based on integration, analysis and evaluation of qualitative and quantitative information and data
- understanding of the ethical, historical, environmental and technological aspects of computer science, and of how science contributes to the social and economic development of society.

The examination will be 2 ½ hours long and will have three sections – A, B and C.

Section A: Short questions that address core computer science topics across the entire specification.

Section B: Practical questions requiring the use of a programming language.

Section C: Questions based on a context but drawn from across different areas of the specification.

Assessment criteria

The student with a high level of achievement:	The student with a moderate level of achievement:	The student with a low level of achievement:
demonstrates a thorough knowledge and understanding of the principles and concepts of computer science, from the whole specification and with few significant omissions	demonstrates a good knowledge and understanding of the principles and concepts of computer science, from many parts of the specification	demonstrates a limited knowledge and understanding of the principles and concepts of computer science

consistently applies knowledge and understanding of the principles and concepts of computer science to problem solving in both familiar and new contexts using appropriate computational thinking	applies knowledge and understanding of principles and concepts of computer science, to problem solving in both familiar and some new contexts using appropriate computational thinking	selects appropriate facts and principles to solve problems concerning familiar material using a limited range of computational thinking
is able to write, compile, test and debug program code in a manner that is almost flawless	is able to write code and can compile, test and debug program code with some errors	only has a limited ability to write code and to compile, test and debug program code
consistently designs, programmes and evaluates computer systems that solve problems, making reasoned judgements about these and presenting conclusions.	designs programmes and evaluates some computer systems that solve problems, making judgements about these and presenting conclusions.	designs programmes that do not solve problems they were designed to. Presents limited evaluation of some computer systems without making judgements about these or presenting conclusions
demonstrates a thorough knowledge and understanding of the ethical, historical, environmental and technological aspects of computer science, and of how science contributes to their personal lives and to the social and economic development of society.	demonstrates a good knowledge and understanding of the ethical, historical, environmental and technological aspects of computer science, and of how science contributes to their personal lives and to the social and economic development of society.	demonstrates a limited knowledge and understanding of the ethical, historical, environmental and technological aspects of computer science, and of how science contributes to their personal lives and to the social and economic development of society.

Table 4 End of course examination assessment criteria

Coursework assessment

The coursework assessment will use practical situations to assess how students design data structures and develop algorithms, integrate ideas, test hypotheses, and explore alternative approaches. It will mirror the structure of the five projects that students complete during the two years of the course, however, the coursework assessment must be carried out individually. Students will not be permitted to work in groups for the coursework assessment.

Each year, in January, the State Examinations Commission will set a task in which students are required to generate a computational artefact based on set criteria outlines by the State Examinations Commission. Each year the Examination Commission will set a date for submission of completed coursework. The time-period for completion of the coursework is six weeks, after which the computational artefact and the report are submitted electronically to the State Examinations Commission for marking.

Assessment criteria

The student demonstrating a high level of achievement:	The student demonstrating a moderate level of achievement:	The student demonstrating a low level of achievement
systematically breaks down problems and filters out unnecessary information and can explain the processes involved. Uses innovative thinking in design and development	identifies problems/things that can be solved. Uses innovative thinking in design and development	engages with limited aspects of the problem. Avoids problems/challenges that have more than one step or part to solving them. May unintentionally overly complicate problems
independently designs, models, tests, debugs and refines solutions (using a test plan and data where appropriate), and chooses an appropriate way to represent solutions	iteratively develops, tests, and debugs solutions	expresses ideas at a basic level. Submits the first working solution as the finished product. Testing, debugging and refinement of solutions is done in a linear fashion.
consistently displays curiosity to exhaustively investigate and analyse a broad range of appropriate problems / solutions	deals with complexity and with open-ended problems	requires a plan and clear expectations or deliverables. Follows instructions and is limited in their self-direction

independently identifies and acts on patterns in problems/solutions. Independently seeks out pre- existing solutions transferring ideas and/or solutions from one problem context to another	adapts existing knowledge or solutions to solve new problems; weighs-up outcomes carefully	application of previous learning to new problems is limited
The student demonstrating a high level of achievement:	The student demonstrating a moderate level of achievement:	The student demonstrating a low level of achievement
celebrates ambiguity and having different interpretations. Compares the performance of interpretations that do the same thing.	shows an ability to tolerate ambiguity in both problems/solutions.	uses pre-learned solutions to attempt to solve new problems. Has difficulty accepting ambiguity in problems or their solutions.

Table 5 Coursework assessment criteria

Assessment programming language

Leaving Certificate Computer Science does not require a specific language, however for the initial years of the subject, Python and JavaScript will be the languages used in the end-of course assessment; this will be reviewed on an on-going basis. There is no restriction in choice of language used in the projects or in the coursework assessment.

Leaving Certificate Grading

Leaving Certificate Computer Science will be graded using an 8-point grading scale. The highest grade is a Grade 1, the lowest grade a Grade 8. The highest seven grades 1-7 divide the marks range 100% to 30% into seven equal grade bands 10% wide, with a grade 8 being awarded for percentage marks of less than 30%. The grades at Higher level and Ordinary level are distinguished by prefixing the grade with H or O respectively, giving H1-H8 at higher level, and O1-O8 at ordinary level.

Grade	% Marks		Grade	% Marks
H1/O1	90-100		H5/O5	50<60
H2/O2	80<90		H6/O6	40<50
H3/O3	70<80		H7/O7	30<40
H4/O4	60<70		H8/O8	<30

Figure 6 Leaving Certificate grading scale

Reasonable Accommodations/Inclusion

This Computer Science specification requires that students engage with practical application of computational thinking on an ongoing basis throughout the course. In addition, the assessment involves a coursework element, which accounts for 30% of the total marks awarded. This emphasis on practical applications may have implications for students with physical/medical/sensory and/or specific learning difficulties. In this context, the scheme of Reasonable Accommodations, operated by the State Examinations Commission, is designed to assist candidates in the Leaving Certificate who have physical/medical/sensory and/or specific learning difficulties.

Appendix I Glossary of action verbs / command terms

Verb	Description
Analyse	study or examine something in detail, break down in order to bring out the essential elements or structure; identify parts and relationships, and to interpret information to reach conclusions
Annotate	add brief notes of explanation to a diagram or graph
Apply	select and use information and/or knowledge and understanding to explain a given situation or real circumstances
Appraise	evaluate, judge or consider text or a piece of work
Appreciate	recognise the meaning of, have a practical understanding of
Brief description/explanation	a short statement of only the main points
Argue	challenge or debate an issue or idea with the purpose of persuading or committing someone else to a particular stance or action
Calculate	obtain a numerical answer showing the relevant stages in the working
Classify	group things based on common characteristics
Comment	give an opinion based on a given statement or result of a calculation
Compare	give an account of the similarities between two (or more) items or situations, referring to both (all) of them throughout
Consider	describe patterns in data; use knowledge and understanding to interpret patterns, make predictions and check reliability
Construct	develop information in a diagrammatic or logical form; not by factual recall but by analogy or by using and putting together information
Contrast	Detect correspondences between two ideas
Convert	change to another form
Criticise	state, giving reasons the faults/shortcomings of, for example, an experiment or a process

Deduce	reach a conclusion from the information given
Define	give the precise meaning of a word, phrase, concept or physical quantity
Demonstrate	prove or make clear by reasoning or evidence, illustrating with examples or practical application
Derive	arrive at a statement or formula through a process of logical deduction; manipulate a mathematical relationship to give a new equation or relationship
Describe	develop a detailed picture or image of, for example a structure or a process, using words or diagrams where appropriate; produce a plan, simulation or model
Determine	obtain the only possible answer by calculation, substituting measured or known values of other quantities into a standard formula
Differentiate	Identify what makes something different.
Discuss	offer a considered, balanced review that includes a range of arguments, factors or hypotheses; opinions or conclusions should be presented clearly and supported by appropriate evidence
Distinguish	make the differences between two or more concepts or items clear
Estimate	give a reasoned order of magnitude statement or calculation of a quantity
Evaluate (DATA)	collect and examine data to make judgments and appraisals; describe how evidence supports or does not support a conclusion in an inquiry or investigation; identify the limitations of data in conclusions; make judgments about the ideas, solutions or methods
Evaluate (ethical judgement)	collect and examine evidence to make judgments and appraisals; describe how evidence supports or does not support a judgement;
	identify the limitations of evidence in conclusions; make judgments about the ideas, solutions or methods
Explain	give a detailed account including reasons or causes

Examine	consider an argument or concept in a way that uncovers the assumptions and interrelationships of the issue
Find	general term that may variously be interpreted as calculate, measure, determine etc.
Formulate	Express the relevant concept(s) or argument(s) precisely and systematically
Group	identify objects according to characteristics
Identify	recognise patterns, facts, or details; provide an answer from a number of possibilities; recognise and state briefly a distinguishing fact or feature
Illustrate	use examples to describe something
Infer	use the results of an investigation based on a premise; read beyond what has been literally expressed
Investigate	observe, study, or make a detailed and systematic examination, in order to establish facts and reach new conclusions
Interpret	use knowledge and understanding to recognise trends and draw conclusions from given information
Justify	give valid reasons or evidence to support an answer or conclusion
List	provide a number of points, with no elaboration
Measure	quantify changes in systems by reading a measuring tool
Model	generate a mathematical representation (e.g., number, graph, equation, geometric figure) for real world or mathematical objects, properties, actions, or relationships
Order	describe items/ systems based on complexity and/or order
Outline	give the main points; restrict to essentials
Plot	a graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables.

Predict	give an expected result of an event; explain a new event based on observations or information using logical connections between pieces of information
Prove	use a sequence of logical steps to obtain the required result in a formal way
Provide evidence	provide data and documentation that support inferences or conclusions
Recognise	identify facts, characteristics or concepts that are critical (relevant/appropriate) to the understanding of a situation, event, process or phenomenon
Recall	remember or recognise from prior learning experiences
Relate	associate, giving reasons
Sketch	represent by means of a diagram or graph (labelled as appropriate); the sketch should give a general idea of the required shape or relationship, and should include relevant features
Solve	find an answer through reasoning
State	provide a concise statement with little or no supporting argument
Suggest	propose a solution, hypothesis or other possible answer
Synthesise	combine different ideas in order to create new understanding
Understand	have and apply a well-organised body of knowledge
Use	apply knowledge or rules to put theory into practice
Verify	give evidence to support the truth of a statement

Appendix II Glossary of terms

Abstraction

Abstractions are formed by identifying patterns and extracting common features from specific examples to create generalizations. Using generalized solutions and parts of solutions designed for broad reuse simplifies the development process by managing complexity.

Data

Data is collected with both computational and noncomputational tools and processes. Students learn how data about themselves and their world is collected and used. Data is collected and stored so that it can be analysed to better understand the world and make more accurate predictions.

Computing systems

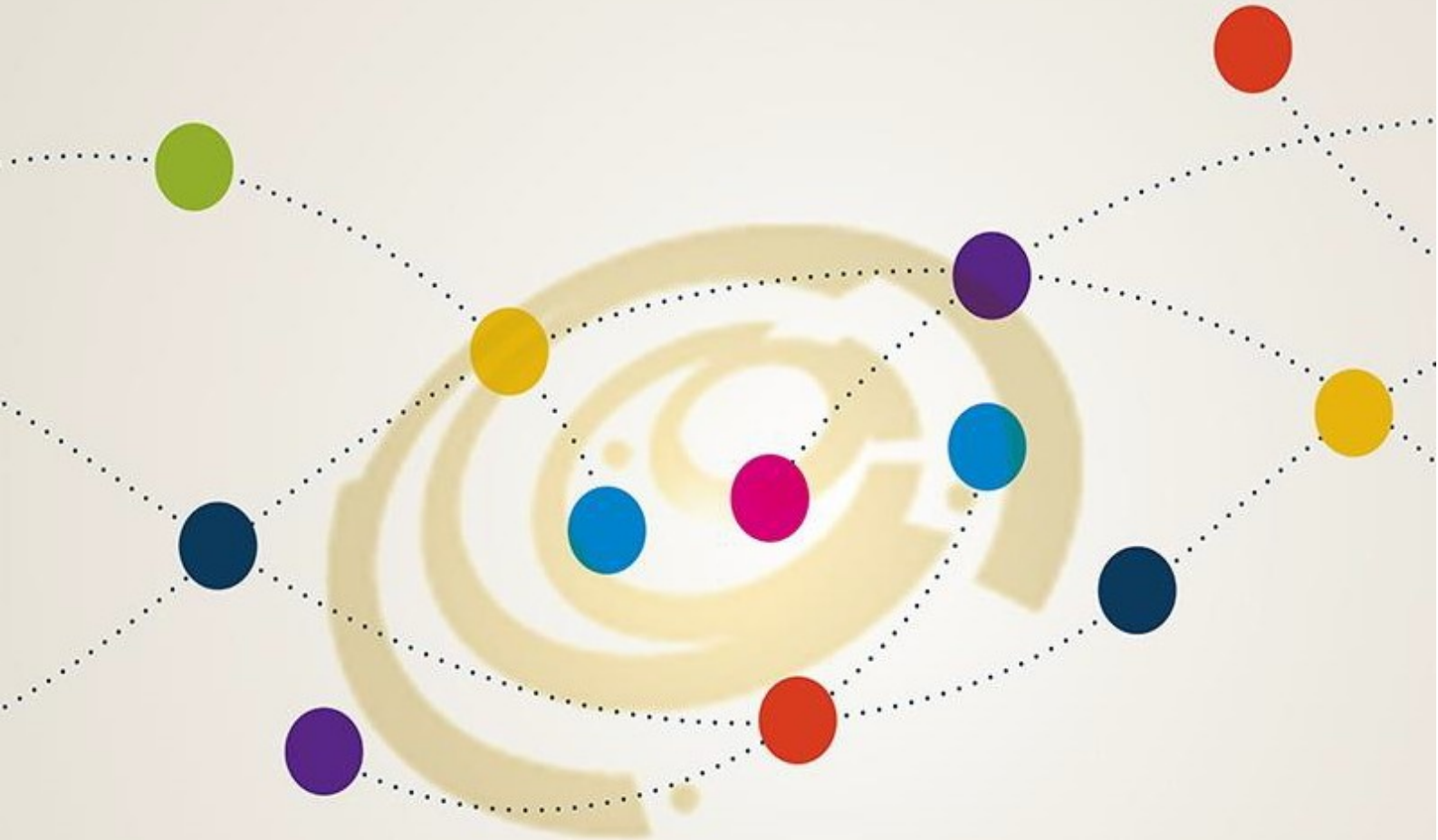
A computing system is the combination of hardware, software, computational processes and networks and users to create a system. Students will develop programming, analysis and design skills combined with the hardware knowledge to create network/Internet/cloud based applications. They will learn how computing devices (such as smart devices, desktop computers and tablets) communicate with each other and the world around them and how to plan and design the infrastructure and systems that allow this to happen.

Algorithms

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. students learn how to read, write, modify and test algorithms, as well as how to evaluate competing algorithms.

Evaluation/testing

Students learn how to identify and remove some errors from computer hardware or software and checks that it does what it is supposed to do. A method often used in computer science is an iterative development process which involves identifying a problem; devising and testing solutions; evaluating the results; and revising and redoing to find the best solution. Central to this process is making mistakes and learning from them to effectively solve new problems in different situations. Students also learn to evaluate the feasibility of using computational tools to solve given problems or sub- problems, such as through a cost-benefit analysis. In their evaluations, students will include factors such as how efficiency effects feasibility or whether a proposed approach raises ethical concerns in their evaluations.



NCCA

June 2017