TRINITY SIXTH FORM ACADEMY Developing Life Ready Learners

Subject Area: Computer Science

Curriculum Intent: Computer Science at Trinity Sixth Form Academy revolves around the main principles of the subject at any level which are to be able to think abstractly, think logically and how to translate real-world problems into computational form. The first two skills are valuable in any subject and can lead to better outcomes at higher levels of study and in careers. Our course is also designed to foster creativity, a prerequisite of a problem-solving discipline. We also place a firm focus on programming skills throughout this course. This has positive implications for success on the course, but again, we are building technical skills that students can use as a springboard into Computer Science degrees and/or the computing and IT job market. In the department we firmly believe all students should leave college as competent programmers as well as being proficient in the theory of the course. We're proud of our strong code of ethics at the college and we strive to take these positive values and frame them in the context of what we learn on the course. This is particularly evident when we look at the two units on *Moral and Ethical Issues* and *Computing Related Legislation* when we encourage students to step outside our subject's theoretical focus and consider the wider world through a computing lens. In doing so, we also overlap with a number of subjects, for example business, economics and even philosophy. This broader thinking is actively encouraged on a weekly basis as we like to consider 'tech' news stories and their wider implications for society and us as computing practitioners. Finally, we consider a common criticism of employers of students with Computer Science qualifications; that they are technically able but poor communicators. This is approached positively as we educate students to see value in what might be called 'geekiness' and how their technical ability can be communicated better to others who are less skilled, especially in a working environment.

Dates	Content	Assessment	Rationale
Year 12 **Term 1 and 2**	**Specification Points (in order of delivery)** 1.4.1 – Data Types 1.2.4 – Types of Programming Language (section (b)) 1.2.3 – Software Development (section (c)) 1.3.1 – Compression, Encryption and Hashing 1.3.3 – Networks 1.5.1 – Computing Related Legislation 1.2.1 – Systems Software 1.4.3 – Boolean Algebra 1.1.3 – Input, Output and Storage 1.4.2 – Data Structures (section (a)) 1.2.2 – Applications Generation (sections (a)(b)(c)(d)) **Programming challenges** Introductory lesson – basic skills Data types – simple conversions between types Strings – basic string manipulation with user input and output Conditions – introducing if statements Iteration – for loops Iteration – while loops Iteration/selection – do while loops Extension challenges are also available on each topic, each week	**PC1:** Exam based on units covered so far, focussing on AO1 and AO2 **PC2**: Exam based on units covered since PC1, focussing on AO1 and AO2 **Weekly**: • Spaced repetition tests based on previous weeks' work • Programming practical testing • Programming theoretical knowledge check **Fortnightly:** • Quiz • Knowledge check	We use the OCR A Level Computer Science specification on our course as it combines in-depth theoretical knowledge with a large body of more practical programming requirements. This is a 'classic' combination that builds nicely from GCSE and prepares students well for Computer Science based degrees. Component 01 is completed in Year 12 as this builds the foundation of theoretical and practical skills that lead naturally into Components 02 and 03. As can be seen, the units are not covered in order of specification points but mixed up throughout the year. This comes from past experience and noting how the specification points fit together best. One eye is always on programming as well, and again this informs the order of delivery of the theory-heavy units and vice versa. Programming starts from week one with a beginner's course in C#. This language has been chosen as the primary one for this course as it features easy access to beginners with a procedural paradigm and later in the year we can move on to the more powerful object-oriented methodologies required for the specification. C# is also being adopted more and more in degree level qualifications and careers in general. Last of all, C# can be used in conjunction with the Unity game engine that we use to create our NEA projects with. Programming is prevalent throughout both papers and the NEA, so where possible theory will be covered using practical tasks, i.e. not just learning about a stack and queue but creating a program which uses them. Programming challenges will test the skills covered in the weeks previous to that challenge. All challenges are linked to a previous exam question, either fully or as an outline. Programming challenges will be closely monitored for completion and will play a more pivotal role in the weekly lesson schedule by having a dedicated lesson each week. Programming challenges will increase in difficulty over the year. Students will be given a scenario to use that will enable them to produce a piece of code. The code will be executed against automated testing, and then automated feedback is presented. When the program has passed the tests, students are provided with a perfect answer so that they can spot inefficiencies in their own code. Students build their confidence through the programming challenges as the automated feedback and error reporting allows them to debug and troubleshoot. One session each week is set aside to introduce the next round of challenges, and to allow students to ask initial questions. Support is also on offer in Achieve sessions.
Term 3 and 4	**Specification Points (in order of delivery)** 1.2.2 – Applications Generation (sections (e)(f)) 1.2.3 – Software Development 1.3.4 – Web Technologies 1.4.2 – Data Structures (sections (b)(c)) 1.1.1 – Structure and Function of the Processor 1.1.2 – Types of Processor 1.5.2 – Moral and Ethical Issues **Programming challenges** Arrays – 1D Arrays – 2D Arrays – Multidimensional Switches Random numbers (including use of external libraries) Reading from a file Writing to a file Functions and Procedures Data structures: tuples, lists, stacks, queues, records	**PC3**: Exam based on units covered so far, focussing on AO1 and AO2 **PC4**: Exam based on units covered so far, focussing on AO1 and AO2 **Weekly**: • Spaced repetition tests based on previous weeks' work • Programming practical testing • Programming theoretical knowledge check **Fortnightly:** • Quiz • Knowledge check	The earlier weeks here involve some theory of programming and move on to a lengthy section on web technologies including a switch in programming skills with the introduction of JavaScript. Again, JavaScript is taught using set challenges with automated testing, feedback and a model solution at the end. The HTML and CSS elements are less automated and also feed into unit 1.5.2 when we like to link with the need to build accessible websites and the moral implications behind doing so. We approach 1.1.1 and 1.1.2 in tandem and then move on to 1.5.2 with a focus on exam technique in the essay-style exam questions used when discussing moral and ethical issues. Our programming lessons now pick up pace and invite more independent learning, whilst still supporting those students who need it. The challenges increase in difficulty but include more scaffolding as required, though students are encouraged to attempt challenges first and seek help only if needed. Paired programming is also used at this stage to increase knowledge throughout the classroom.
Term 5 and 6	**Specification Points (in order of delivery)** 1.2.3 – Software Development (sections (a)(b)) 1.3.2 – Databases 1.2.4 – Types of Programming Language (section (a)(e) theory then practical) 1.2.4 – Types of Programming Language (section (c)(d)) **Programming challenges** Object-oriented programming **NEA preparation** The final four weeks of term	**PC5**: Full paper covering all of Component 01 **Weekly**: • Spaced repetition tests based on previous weeks' work • Programming practical testing • Programming theoretical knowledge check **Fortnightly:** • Quiz • Knowledge check	1.3.2 – Databases is the final, non-programming unit covered in the year. However, this is quite a practical topic that is taught and assessed using a variety of online tools as well as traditional written work for the more theoretical areas such as Entity Relationship Diagrams. The final parts of 1.2.3 and 1.2.4 are taught and overlap well with the move to object-oriented techniques in our programming sessions. The topic of OOP is looked at in great detail both theoretically and practically as it makes up such an important part of both exams. This then leads us into the NEA preparation weeks where again, OOP is prominent. These four weeks act as an introduction to Component 03 as a whole and digs deeper into the practical aspects of what we will be doing, such as the software we will be using. It all builds towards summer work that students can carry out ready for a full assault on the component in September.

Year 13 **Term 1 and 2**	**NEA** Weeks 1 – 4 **Specification Points (in order of delivery)** 2.2.1 – Programming Techniques **NEA** Weeks 5 – 6 **Specification Points (in order of delivery)** 2.1.1 – Thinking Abstractly 2.1.2 – Thinking Ahead 2.1.3 – Thinking Procedurally 2.1.4 – Thinking Logically 2.1.5 – Thinking Concurrently 2.2.2 – Computational Methods **NEA** Week 7	**PC1**: Half of Component 01 **PC2**: Half of Component 01 **Weekly**: • Spaced repetition tests based on previous weeks' work • Programming practical testing • Programming theoretical knowledge check **Fortnightly:** • Quiz • Knowledge check	Year 13 starts as we left off in Year 12 as we begin the NEA properly. Although we have full weeks on the NEA, the implication is that students work on it across all weeks in their Independent Study time too. This is aided by extra time being available outside normal class times for them to complete work on specialist software in the Computer Science classroom. The NEA is a programming project that meets the criteria set out for Component 03. We've chosen to build a video game for the project which has proven to engage students very well along with deepening their programming skills. We use the Unity game engine for this, allied to C#. This combination is laid out as one of the examples for Component 03 in the OCR specification. Units 2.1.1 – 2.2.2 involve the theoretical thinking required to work well as a software developer and naturally tie in with the NEA. They also build on the foundation of programming skills learned in Component 01 in Year 12.
Term 3 and 4	**NEA** Week 8 **Specification Points (in order of delivery)** 2.3.1 – Algorithms **NEA** Weeks 9 – 11 **Specification Points (in order of delivery)** 2.3.1 – Algorithms **NEA** Final week and hand in	**PC3**: Based on units completed so far in Component 02 **PC4**: Trial exams – both papers **Weekly**: • Spaced repetition tests based on previous weeks' work • Programming practical testing • Programming theoretical knowledge check **Fortnightly:** • Quiz • Knowledge check	NEA work continues and comes to its conclusion. The theoretical 2.3.1 unit is taught and spread over a number of weeks. Exam technique becomes more involved, building especially towards the trial exams.
Term 5	**Revision** Week 1: • 1.1.1 - Structure and function of the processor • 1.1.2 - Types of processor • 1.1.3 - Input, output and storage • 1.2.1 - Systems Software • 1.2.2 - Applications Generation Week 2: • 1.2.3 - Software Development • 1.2.4 - Types of Programming Language • 1.3.1 - Compression, Encryption and Hashing • 1.3.2 - Databases • 1.3.3 - Networks Week 3: • 1.3.4 - Web Technologies • 1.4.1 - Data Types • 1.4.2 - Data Structures • 1.4.3 - Boolean Algebra • 1.5.1 - Computing related legislation • 1.5.2 - Moral and ethical Issues Week 4: • 2.1.1 - Thinking abstractly • 2.1.2 - Thinking ahead • 2.1.3 - Thinking procedurally • 2.1.4 - Thinking logically Week 5: • 2.1.5 Thinking concurrently • 2.2.1 Programming techniques • 2.2.2 Computational methods • 2.3.1 Algorithms Remaining time – revision as requested by students.	**External A-Level exams May/June** **Weekly**: • Spaced repetition tests based on previous weeks' work • Programming practical testing • Programming theoretical knowledge check **Fortnightly:** • Quiz • Knowledge check Working on timing of questions Emphasis on longer exam questions and techniques behind answering them	Although a fairly strict plan of revision is given here, this has some flexibility dependent on students' needs. Work is based around building revision resources for each topic along with discussing key terminology, both in terms of understanding and how that should translate into answering exam questions. We consider past paper questions extensively. Longer essay style questions are considered and techniques introduced to answer those, including timing.