Cardinal Newman Catholic School
Department of Computer Science and Creative iMedia

INTENT: Curriculum Overview: **Year 11 Computer Science**

A learner in Year 11 will know: what robust programming is, translators and facilities of languages computational logic and data representation and how to apply the content through exam techniques. The will know about the knowledge and understanding gained in Component 1; This knowledge and understanding will be applied to computational thinking. Students will know about algorithms, how to design them, correct them and identify errors in algorithms. They will know about programming techniques, how to produce structured code, refine code and spot errors.			**A learner in Year 11 will be able to:** demonstrate the relevant and comprehensive knowledge and understanding of fundamental concepts in programming techniques, computational thinking, algorithms and different types of algorithms. Added to this the knowledge they have gained in this unit along with that in component 1, students will become well equipped to perform well in the two exam papers.		
11.1 Topic 2.1 Algorithms 2.2 Programming fundamentals (Recap 1.1, 1.2 & 1.3)	**11.2 Topic** 2.3 Producing Robust Programs (Recap 1.4, 1.5 & 1.6)	**11.3 Topic** 2.5 Programming Languages and IDEs	**11.4 Topic** Revision and past paper practise	**11.5 Topic** Revision and past paper practise	**Exams**

Term 1	**11.1 Topic** 2.1 Algorithms 2.2 Programming fundamentals	**11.2 Topic** 2.3 Producing Robust Programs	Assessment
	2.1 Algorithms **Knowledge:** Students will gain an understanding of computational thinking and be able to describe the terms: abstraction, decomposition and algorithmic thinking. They will be able to design, create and refine algorithms using the following processes: pseudocode, flowcharts, Python code. They will be able to interpret algorithms using trace tables and understand the methods of common searching (linear & binary) and sorting (bubble, merge & insertion) algorithms. **Skills:** • Identify where abstraction, decomposition and algorithmic thinking have been used • From a given scenario, design, create or refine algorithms using pseudocode, flowcharts and Python code • Apply a trace table to a given algorithm • Apply the searching and sorting algorithms to a given data set • Discuss the merits and drawbacks of searching and sorting algorithms **2.2 Programming fundamentals** **Knowledge:** Students will have an understanding of the three programming constructs: sequence, selection & iteration. Students will	**Knowledge:** This topic requires students to gain an understanding in how to ensure their programs are fully functional and are less prone to error. They will learn defensive design techniques such as: anticipate misuse, input validation and maintainability skills. They will also be able to spot both syntax and logic errors and test programs using normal, boundary & erroneous data. **Skills:** • Anticipate the misuse of programs by identifying how people may use a program • Use input validation methods such as: range check, presence check, length check and lookup tables. • Use maintainability techniques such as: sub-programs (procedures & functions), use suitable variable names, indentation and commenting. • Identify syntax & logic errors in code • Create testing tables using normal, boundary and erroneous data **Formative Assessment:** At the end of this topic knowledge will be tested with a pitstop assessment	Knowledge coverage: ○ Abstraction ○ Decomposition ○ Algorithmic thinking ○ Pseudocode ○ Flowcharts ○ Python programming fundamentals **Assessment style/questions:** Exam style questions, combination of short written answers requiring students to state, explain, describe analyse and compare.

	understand the use of common data types; integer, float, character/string and Boolean. They will be able to produce/refine code to a given scenario that includes use of variables, constants, inputs, outputs and assignments, common arithmetic operators and the use of sub-programs (procedures & functions). They will be able to use string manipulation methods and file handling operations and be able to interpret data in 1D and 2D arrays. **Skills:** • Design, create and refine code using all of the programming techniques outlined in the knowledge section **Formative Assessment:** At the end of this topic knowledge will be tested with a pitstop assessment		
Term 2	**11.3 Topic** 2.5 Programming Languages and IDEs	**11.4 Topic** Revision and past paper practise	Assessment
	Knowledge: In this topic students will learn about different types of programming languages and be able to explain the differences between high- and low-level languages. They will be able to explain the need for translation, and discuss the merits and drawbacks of compilation and interpretation. They will also be able to explain the features of an IDE that aid programmers. **Skills:** • Explain the difference between high- and low-level languages • Explain translation including the different methods • Explain the following features of an IDE: editor, error-diagnostics, run-time environment, translators **Formative Assessment:** This topic, along with all topics covered during this course, will be assessed in the first round of year 11 mock assessments. This will be a past exam paper.	The remainder of year 11 will be spent preparing students for the summer exams in May/June. Lessons remain structured with topics being re-taught, students will have exam questions to answer that have come from previous years papers and/or created by their teacher. Analysis of exam papers since the beginning of the specification allows targeted revision sessions to be purposeful, resourceful and well planned.	Knowledge coverage: ○ High- and low-level languages ○ Need for translation ○ Features of an IDE **Assessment style/questions:** Past exam papers

Cardinal Newman Catholic School
Department of Computer Science and Creative iMedia

Term 3	**11.5 Topic** Revision and past paper practise	**11.6 Topic** **EXAM**	Assessment
	The remainder of year 11 will be spent preparing students for the summer exams in May/June. Lessons remain structured with topics being re-taught, students will have exam questions to answer that have come from previous years papers and/or created by their teacher. Analysis of exam papers since the beginning of the specification allows targeted revision sessions to be purposeful, resourceful and well planned.	The remainder of year 11 will be spent preparing students for the summer exams in May/June. Lessons remain structured with topics being re-taught, students will have exam questions to answer that have come from previous years papers and/or created by their teacher. Analysis of exam papers since the beginning of the specification allows targeted revision sessions to be purposeful, resourceful and well planned.	**Assessment style/questions:** Past exam papers