Transition to A Level Computer Science

2022 Start

St John Bosco College

“it is not enough for children to be loved, they must know that they are loved”

#Believe And Achieve

Parkham Street, Battersea,London SW11 3DQtel: 0207 924 8310 fax: 0207 738 1867e: info@sjbc.wandsworth.sch.ukw: www.sjbc.wandsworth.sch.uk

Contents

Introduction

The aim of this booklet is to ease you into A level Computer Science. The course we follow at Saint John Bosco College is OCR Computer Science. There is a link to the specification on this page. There is also some recommended reading. Some of these are books that could support you throughout the year.

The jump from GCSE to A level is huge and to be a successful Computer Scientist at A level it is essential you work on the key concepts such as Data representation, Data Structures and Searching and Sorting Algorithms. Have a go at the activities in this booklet. Answers will be provided for you to check your understanding. These activities also extend into KS5 content to give you a bit of a challenge. Go to the useful websites page to support you with these activities. Once you have completed the transition activities, start to think about what programming project you would like to carry out and complete the analysis section.

You are expected to complete all work and bring it to your first lesson of Computer Science in September. If you are new to Python or not a very strong programmer, please email me for some additional resources to build up your skills in this area.

OCR Computer Science H446 – Specification
This is a two-year, linear A-Level course which will be formally assessed at the end of the two years.
The content of this A Level in Computer Science is divided into three components:

• **Computer systems component (01)** contains the majority of the content of the specification and is assessed in a written paper recalling knowledge and understanding.

• **Algorithms and programming component (02)** relates principally to problem solving skills needed by learners to apply the knowledge and understanding encountered in Component 01.

Computer Science Teacher: Mrs M Perrineau-Daley
Email address: mperrineau-daley@sjbc.wandsworth.sch.uk

Recommended Reading

The following is a list of suggested texts that may help you with your studies of A Level Computer Science, it is not an exhaustive list.

OCR AS and A Level Computer Science:
PM Heathcote and RSU Heathcote
ISBN: 978-1-910523-05-6
You **will need** to purchase this book for the course. It is available on Amazon

This textbook covers Component One and Two of the A Level course and also has a section on Completing Component Three.

Tackling A Level projects in Computer Science OCR H446:
Ceredig Cattanach-Chell
ISBN: 978-1-910523-19-3
Available on Amazon

This is an extensive how-to guide on completing the programming project. Starting with how to choose a project to work on and then how to write about it. It has been written by one of the OCR team who has
been heavily involved with the creation of the GCSE and A-Level Computer Science specifications.

Essential algorithms for A Level Computer Science:
D Hillyard and C Sargent
ISBN: 978-1-794359-42-0
Available on Amazon

This book details all of the algorithms and data structures that you need to know for the A Level course. Each algorithm has visual representations, pseudocode examples and working Python and Visual Basic code examples.

Useful Websites

Craig n Dave Videos - https://www.youtube.com/c/craigndave/

These videos are specifically tailored to the course and cover each element in the specification. They will be used for flipped learning before each lesson.

101 Computing - https://www.101computing.net/category/a-level-concepts/
Lots of facts and interactives to help you to understand some of the different topics covered in the A Level course. There are also sections for the GCSE content as well if you want/need a refresher.

W3schools - https://www.w3schools.com/
Tutorials and examples of code used for various different aspects of web development, such as HTML, CSS, JavaScript, SQL. Also has a "Try it" editor which allows you to write code and then see it in action within your web browser.

PacktPub Free Learning - https://www.packtpub.com/free-learning
A free eBook every day that focuses on different programming languages and projects that can be completed with them. They often have books that cover Web Design, Artificial Intelligence, and Games Design to list but a few.

Crash Course Computer Science - https://www.youtube.com/watch?v=O5nskjZ_GoI
A YouTube playlist created by Carrie Ann Philbin (Director for Education at the Raspberry Pi Foundation) that gives an explanation of what Computer Science is, ranging from how it began to where it is heading.

Crash Course Artificial Intelligence - https://www.youtube.com/watch?v=a0_Io_GDcFw
A YouTube playlist that has been based on a University-level curriculum. It looks at the principals of Artificial Intelligence and Machine Learning and discusses their applications.

Transition Activities

Data Types, Data Structures and Algorithms

Activity 1

Converting between denary, binary and hex

No.	Denary	Binary	Hex	Binary value plus 00011110
1	1			
2	5			
3	10			
4	22			
5	40			
6	77			
7	91			
8	121			
9	144			
10	168			
11	170			
12	200			
13	211			

Activity 2

Create a program that analyses a passage of text from a file and then counts:

- How many words
- The average length of a word
- How many times each word occurs
- How many words start with each letter of the alphabet?

The aim of this exercise is to test your ability to develop algorithms.

Binary Truth Tables

1. Write the truth tables for the expressions:

NOT (A AND B)

and ((NOT A) OR (NOT B))

2. What do you notice about these tables?

3. Design and create a program to output the value of **a** after the statement:

   IF (a < b) OR (b < c) THEN a = b

   has been executed.

4. Decide on suitable test data for this program giving a reason for each combination of values for **a, b** and **c** , give your expected result and the actual result for each.

Values	**Reason**	**Expected**	**Actual**

RPG character simulator

Planet of Fight craft wants you to build character classes for their new game.
Each character will have the following things:

- Name
- Type (Barbarian, Elf, Wizard, Dragon, Knight)
- Health
- Power
- Special attack power
- Speed

All characters start with 100 health.
Different creatures have different power ratings (B: 70, E: 30, W: 50, D: 90, K: 60)
Different creatures have different special attack power ratings (B: 20, E: 60, W: 70, D: 40, K: 10)
Different creatures have different speed ratings (B: 50, E: 10, W: 30, D: 50, K: 60)

Tasks

1. Generate a random name: en-da-fu and el-kar-tuk could be names, so you could make a name generator which sticks together three syllables from 'word banks'.
2. Create the generic character class. Test to see if you can create multiple characters.
3. Create subclasses corresponding to different types of creature (B, E, W, D & K).
4. Make a program that randomly generates 10 of these creatures to add into a list.
5. Make a method in the character class that enables printing out of each character's stats to the console.
6. Create a menu system that lets you add and delete characters and print out the list until you are happy with the team.
7. Create methods to let you edit any character's stats and add this to your menu system.
8. Create a way to save your team to a file and load it up again if needed.

How to evidence your work.

1. Please save your code in a simple text document such as notepad.
2. Make sure you have included comments in your code

When you have completed your work email to me at: **mperrineau-daley@sjbc.wandsworth.sch.uk**

Types of Programming Language

Note: The following activities are designed to be a bridge between what is studied at A Level and typical content that may be covered at University. Please feel free to have a go at these tasks but it is not expected that you complete them.

Activity 1a: Average of an array
Create a procedural program that can be described in structured English as follows:
The 'Average of an Array' program

Step 1: User repeatedly enters numbers into an array.
Step 2: Array average is calculated by finding the sum of the array and the number of elements in it.
Step 3: The result of the calculation is output.
Step 4: User is asked if they want to repeat the program.

Activity 1b: Redo your program from 1a using functions (subs that return values) for all procedures except main()

Activity 1c: How would you carry out unit tests on your procedures?

Activity 2a: Write a program that matches the following structure with comments that explain your code:

Sub1
Sub2
Function1
Function2
Main

Activity 2b: If you were not constrained by this structure, how would you implement the same program? Can you think of a better structure?

Activity 2c: A Learner wrote a program with this very clever line in the procedure 'main':

"show_km(convert(validate(read_miles ( ))))"

Explain how the Learner was able to do that and complete this program by creating the procedures necessary to make this line work.

Activity 2d: Rewrite the program in the imperative procedural paradigm with comments.

Activity 2e: Name the state variables.
What is their usefulness?
How could using state variables create problems?
Provide an example of this situation.

Activity 2f: List the differences between your two programs.

Stretch exercises:

Activity 2g: Modify both versions of your program to ask the user the direction of the unit conversion, e.g. ‘miles to km’ or ‘km to miles’.

Activity 2h: Modify both programs to ask if a user wants another distance converted. Which paradigm was easier to modify?

Programming Project – Analysis

In Year 13 you will be creating your own piece of software for your Programming Project. This is worth 70 marks. This is equivalent to 20% of your final A Level marks.
To help you get a head start with this, over summer you need to think about what you would like to do for your programming project and who your end user will be.
You will then need to complete the Analysis phase of the project by documenting:

- What the Project is.
- Who your end-user will be.
- Anyone else who might be affected by you project.
- Research existing/similar programs
- Identify requirements and success criteria.
- Discuss any limitations your solution might have.

Use the mark scheme and guidance below to help you with this.

AO 2.2 Analysis (maximum 10 marks)			
1–2 marks	**3–5 marks**	**6–8 marks**	**9–10 marks**
The candidate will have:			
• Identified some features that make the problem solvable by computational methods. • Identified suitable stakeholders for the project and described them and some of their requirements. • Identified some appropriate features to incorporate into their solution. • Identified some features of the proposed computational solution. • Identified some limitations of the proposed solution. • Identified some requirements for the solution. • Identified some success criteria for the proposed solution.	• Described the features that make the problem solvable by computational methods. • Identified suitable stakeholders for the project and described how they will make use of the proposed solution. • Researched the problem looking at existing solutions to similar problems identifying some appropriate features to incorporate into their solution. • Identified the essential features of the proposed computational solution. • Identified and described some limitations of the proposed solution. • Identified most requirements for the solution. • Identified some measurable success criteria for the proposed solution.	• Described the features that make the problem solvable by computational methods and why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them and how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems identifying and describing suitable approaches based on this research. • Identified and described the essential features of the proposed computational solution. • Identified and explained any limitations of the proposed solution. • Specified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified measurable success criteria for the proposed solution.	• Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research. • Identified the essential features of the proposed computational solution explaining these choices. • Identified and explained with justification any limitations of the proposed solution. • Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified and justified measurable success criteria for the proposed solution.

0 marks = no response or no response worthy of credit.

In this analysis phase you need to make sure to include all the following sections:

1. Defining the problem & the Stakeholders

Start by giving a brief background to the problem. Answer the questions:
What is the company?
What does the company do?
Who are the stakeholders / end users?
What problem do they have?
How will they make sure of your proposed solution and why is it appropriate to their needs?

This initial description of the problem should be no more than a couple of paragraphs.

If you are writing a computer game, give a description of the type of game it is with a very brief explanation.
Explain who will play the game and on what platform. Identify the key requirements outside the actual game play itself, e.g. in the case of a mobile phone game: easy to pick up and put down, pause at any point and continue later.

If the problem has already been solved, pretend you are solving it for a new platform or user. e.g. with space invaders, perhaps it is a mobile phone version or a new twist on the old concept.

Note about stakeholders: Make sure you clearly name all the stakeholders / users for your system. These must be actual named individuals that you can have regular contact with as they will be required to give you feedback and interviews throughout the development of your project. You can have more than one stakeholder / user. For example, if you are creating a Maths revision utility for Year 11's then you would clearly have two users, Maths teachers and Year 11 students. They will both be able to give you requirements and feedback from their different perspective.

It is also acceptable to have chosen a "persona", someone who personifies the typical user for your chosen system.
This will be most likely if you choose to make a game. Decide who your game is targeted at e.g. "Teenagers into mobile gaming" and then choose a named person from this target group who you will be able to have regular contact with to act as your stakeholder / end user?

Make sure in this section to not just simply list our users / stakeholders. For top marks you must make sure to explain how they will make sure of your proposed solution and explain why it is suitable for their needs.

2. Justification of how the problem can be solved by computational methods

You must fully justify how the solution you wish to program can be solved by computational methods. These are all the methods you will be studying for Unit 2 and include:

- Thinking Abstractly & Visualisation
    - How will your problem simplify reality? If you are producing a game, simulation, training aid, booking system etc what detail IS important and what details from reality will you ignore or omit?
- Thinking Ahead
    - What data / inputs will be required for your solution to work?
- Thinking Procedurally & Decomposition
    - Can your problem be easily broken down and tackled in smaller chunks?
- Thinking Logically
    - Will your problem have obvious decisions points for branching or repetition (looping)?
- Thinking Concurrently
    - Will there be any parts of your problem which could be solved or could happen at the same time?

3. Research

In this section you are describing the problem. With a game, take this approach to the write up:

1. Initial research: start by identifying a similar game (perhaps from the internet) and describe the mechanics of how it plays.
2. Form a set of questions to ask the user about how your game should look, sound and play. Document the user responses to these questions. (See Note 1) e.g.

   Q: What does the player control in the game?

   A: The player controls a spaceship that can move left and right at the bottom of the screen.
3. Deliberate on the answers you are given and the initial research. This will inform the proposals.
4. Propose a solution to the problem by describing each element of the game in detail. You can have mock-ups of the graphics from a drawing application at this stage. (See Notes 2 and 3)
5. Get a response from the user about whether this meets their expectation.
6. Get an agreement from the user.

Note 1: You need to conduct an interview and/or observation of at least one existing system to know the details of what you need to know to make the program later. Keep records of the questions and observations you make, together with answers to questions.

Note 2: You need to discuss in detail exactly what the system is going to do, but not how it is going to do it. This is not about design or algorithms, it is about the requirements. Here we are focusing on the what, not the how. Detail is very important in this section in your descriptions of the system.

Note 3: Consider a typical space invaders game. You would need to discuss that the player controls a ship. The ship can move left or right inside a fixed plane at the bottom of the screen. The ship can fire one bullet at a time. There cannot be more than one bullet from the player on the screen at the same time. The objective is for the player to shoot all the invaders. The invaders start towards the top of the screen and move from left to right together in initially 5 rows of 12 invaders. When the right-most invader reaches the right edge of the screen, all the invaders move down a little on the screen and start moving from right to left. When the left-most invader reaches... etc.

'Leave no stone unturned'. Your analysis should include sufficient detail so if you were to get a programmer to read the analysis, there is nothing more they would need to ask before making the solution. Of course, the really fine details may not be entirely known and will be picked up in the development process. For example, the speed at which the ship moves across the screen. You would need to play the game to know what feels right. That is unlikely to be known at the analysis stage and the necessary dialogue between you and a user will gain you marks in the design section later.

You should write your analysis as if you were having a discussion with a user. For example,

"The intended audience for the game is..." "I am using my teacher as a representative for that audience." "I discussed the requirements of the game with..." "It was suggested to me that..." Whatever the problem, it will always have a target audience and therefore an identifiable user which you should be discussing requirements with and keeping records of these discussions.

4. Features of your proposed solution

In this part you should make sure to clearly explain each of the features of your proposed solution. How you choose to do this is up to you, however, look carefully through your research and analysis and make sure you have not missed anything.

In this section you should also identify any limitations of your proposed solution. It will, by the nature of an A Level project be limited. If it is a game, what will it not do, be realistic. This is a good time to flag up desirable features that will not be included in the solution (you can revisit this again when you write your evaluation at the end).

5. Hardware & Software requirements

You should discuss the hardware and software required to run your program. e.g. an IBM compatible PC with x processor, y memory and z hard disk space running the Visual Basic runtime libraries? Find out the necessary spec to run the development environment i.e. VB or Access on a computer.

If any additional software is required to run your solution or if your solution is only intended to work with specific versions of software this needs to be identified here.

6. Success criteria / Requirements specification

As a summary of the analysis, create a numbered point list or table of the exact and actual success criteria / requirements. Call this the "Success Criteria / Requirements Specification."
Avoid requirements that cannot be measured. e.g. "It must be easy to use" is too vague. "The user should be able to find a product within 20 seconds" is better. "A player scores 50 points for each invader." Remember, specific and measurable. It should contain numbers. Numbers of records, users, invaders, points etc.