SIXTH FORM CENTRE
UNIVERSITY COLLEGE BIRMINGHAM

A-level Computer Science

Summer Transition Work

A-level Computer Science

Computer Science Summer Work

The purpose of this work is to enable you to start your life at UCB6 in the best way possible. In preparation for your study of Computer science there is some foundational knowledge that will be important for you to know. If you have not studied computer science before you will need to work hard to use this summer work to catch up. Please complete the following tasks. You will be required to show evidence of your learning in September.

Task 1: Input / Output / Storage

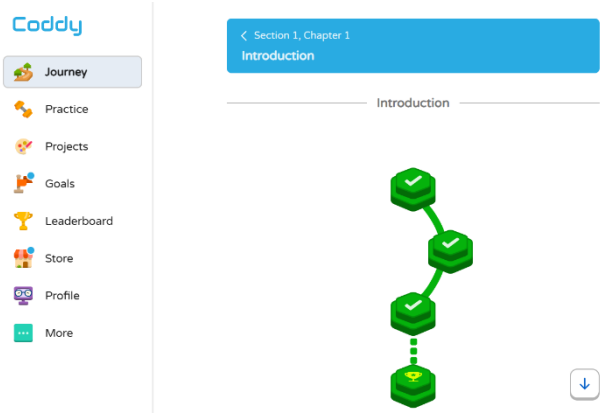
Input, output and storage are something that you will have covered in KS3 and is our starting point for learning at A level.

Watch the following videos on IOS and make notes.

Answer the following questions:

Bring your marked work with you to your first lesson back in September

Task 2: Python programming



Coddy is a Duolingo style application that helps you learn to code. It is available as both a website and a mobile application on both IOS and Android. Regardless of previous experience, please sign up for a free account using your email address. Remember to select Python as your learning language.

https://coddy.tech/

Complete everything up to: Chapter 1: Section 5.

Task 3: Maths

Maths is a core part of Computer Science therefore you must have a good understanding of basic principles. **Answer the following questions and bring them back in your first lesson back.**

1. What is the remainder when 27 is divided by 6?
2. $A = 2 \times 10^3$. $B = 4 \times 10^2$ How many times larger is A than B?

3. A photo grid starts at coordinate (0,0) in the top-left corner. If you move 5 pixels to the right and 5 pixels down, what is the coordinate of that pixel?
4. List all the common factors between 16 and 64.

Task 4: Logic

When computer scientists design systems, they rarely work with simple "right or wrong" scenarios. Instead, they write algorithms that must constantly balance competing real-world priorities. Without logic computers would not function correctly. Logical thinking is a cornerstone of computational understanding and forms a large part of the way a computer works. It is imperative that you can think logically through a problem, explain your thought process, and trace how data changes state through a system - a fundamental skill that underpins everything from designing basic algorithms to constructing complex Boolean circuits.

Imagine you are working as a Systems Architect for a tech firm developing the automated navigation brain for a next-generation driverless ambulance. The vehicle must make split-second decisions about its speed, route, and driving style based on live telemetry data.

Configure the software's Priority Weights. You have a total budget of 100 Priority Points to distribute across four core system rules.

System Rules:

- **Patient Stability:** Focuses on smooth driving, avoiding harsh braking or cornering to keep internal medical monitors steady.
- **Speed of Arrival:** Focuses on getting to the emergency room as quickly as possible, utilizing shortcuts and maximum legal speeds.
- **Pedestrian & Public Safety:** Focuses on minimizing risk to people outside the vehicle, heavily reducing speed near crossings or busy town centres.
- **Asset Preservation:** Focuses on avoiding mechanical damage to the expensive ambulance vehicle and its onboard medical equipment.

Constraints:

1. You must allocate **exactly 100 points** across the four rules in total.
2. You **cannot** give two rules the same number of points (every value must be unique).

Task:

1. Allocate your 100 points across the four directives (e.g., A: 40, B: 30, C: 20, D: 10)
2. Write a short paragraph explaining the rational thought process behind your highest priority and your lowest priority.
3. Imagine the ambulance is carrying a critical patient who needs surgery within 5 minutes, but the route passes a busy school zone at 3:30 PM. Explain how *your specific points allocation* would force the algorithm to behave in this exact scenario.

Answer the questions and be prepared to justify them in a class debate during the first week back.

Wider Reading

This list is not exhaustive and is just a small portion of encouraged reading.

- BBC Click - http://www.bbc.co.uk/programmes/n13xtmd5
- MT News - http://news.mit.edu/topic/computers
- Phys.org - https://phys.org/technology-news/computer-sciences/
- Wikibooks - OCR A-Level Computing H446

Useful Websites

This list of websites will come in handy throughout your next two years and beyond – it is worth saving them for your own support.

- Craig'n'Dave - YouTube
- OCR A-Level Computer Science Revision - PMT (physicsandmathstutor.com)
- Dashboard | HackerRank
- https://coddy.tech/journeys/python/fundamentals
- https://tinypythonprojects.com/
- https://isaaccomputerscience.org/topics/a_level?examBoard=ocr&stage=a_level#ocr

Final notes

Students say that they wish they started revising their notes sooner. It is important to make sure that you are consolidating your notes and ideas after every lesson. Ensure that you are utilising the time given in centre to secure your knowledge or come and find me if you are stuck.

Programming, particularly in python is not only important for Unit 2 but will make up most of your project work (worth 20%). You must be practicing your programming every week.

Be prepared to learn – this means having accessed the canvas page, completed your homework, and come to lesson on time with all your equipment.

Respond appropriately to any feedback given. You will constantly be receiving feedback so make sure that you are acting upon so that you can succeed.

Read around the subject. The more knowledge of news, technology, and the world that you have the more you will be able to apply learnt computing information.

Lastly, enjoy it. Computer Science isn't just for now, it's for ever! So, enjoy it, don't be afraid to get things wrong and keep at it!