COMPUTER SCIENCE

Computer Science bridging work

Task 1: Understanding Basic Data Types (Approx. 2 Hours)

Objective: To understand what data types are, why they are important, and the common types used in computing.

Activities:

1. **Research & Reading (1 hour):**
    - **What are Data Types?** Begin by researching what "data types" mean in the context of computer science. Why do computers need to know the type of data they are handling?
    - **Common Data Types:** Investigate the following fundamental data types. For each, understand its purpose, the kind of data it stores, and typical examples:
        - **Integer:** Whole numbers (e.g., 5, -100, 0)
        - **Real/Float:** Numbers with decimal points (e.g., 3.14, -0.5, 99.99)
        - **Boolean:** True/False values
        - **Character:** A single letter, number, or symbol (e.g., 'A', '7', '$')
        - **String:** A sequence of characters (e.g., "Hello World", "Computer Science")
    - **Resources:** Use online resources like BBC Bitesize, Computer Science education websites, or introductory programming tutorials (focus on the concepts, not the code syntax yet).
2. **Application & Reflection (1 hour):**
    - **Real-World Examples:** For each data type listed above, think of at least three real-world scenarios or pieces of information where that data type would be the most appropriate choice. For example, "The number of students in a class" would be an Integer.
    - **Data Type Selection:** Imagine you are designing a simple program for a library. For each piece of information below, identify the most suitable data type and explain why:
        - Book Title
        - Number of copies of a book
        - Price of a book
        - Is the book currently available (Yes/No)?
        - The first initial of the author's name
        - The ISBN (International Standard Book Number)

Task 2: The Binary Number System (Approx. 3 Hours)

Objective: To understand how computers represent numbers using binary and perform basic conversions.

Activities:

1. **Research & Reading (1.5 hours):**
    - **Why Binary?** Discover why computers use the binary (base-2) system, consisting only of 0s and 1s, instead of the denary (base-10) system we use daily.
    - **Bits and Bytes:** Understand the terms "bit" (binary digit) and "byte" (typically 8 bits).
    - **Denary to Binary Conversion:** Learn how to convert denary (base-10) numbers into 8-bit binary numbers. Practice with numbers up to 255.
    - **Binary to Denary Conversion:** Learn how to convert 8-bit binary numbers back into denary.
    - **Binary Addition:** Understand the basic rules of binary addition (0+0=0, 0+1=1, 1+0=1, 1+1=10 [0 carry 1]). Practice with simple 4-bit additions.
    - **Introduction to Hexadecimal (Optional but Recommended):** Briefly research hexadecimal (base-16) and its relationship to binary (e.g., how 4 binary bits can be represented by one hex digit). You don't need to master conversions, just understand its purpose.
2. **Practice Exercises (1.5 hours):**
    - **Conversion Practice:**
        - Convert the following denary numbers to 8-bit binary: 10, 42, 128, 200, 75.
        - Convert the following 8-bit binary numbers to denary: 00001101, 00101010, 10000000, 11110000, 01010101.
    - **Binary Addition:**
        - Perform the following binary additions (show your working):
            - 0101 + 0011
            - 1010 + 0101
            - 1111 + 0001
    - **Self-Correction:** Use online binary converters to check your answers.

Task 3: Core Programming Concepts (Approx. 4 Hours)

Objective: To grasp fundamental programming constructs that are common to almost all programming languages. You will use pseudocode (a simplified, language-agnostic way to describe algorithms) and flowcharts.

Activities:

1. **Variables and Constants (0.5 hours):**
    - **Research:** Understand the difference between a variable (a named storage location whose value can change) and a constant (a named storage location whose value remains fixed).
    - **Pseudocode Practice:** Write pseudocode to declare a variable called score and initialize it to 0. Then, add 10 to score. Declare a constant called PI and set its value to 3.14159.
2. **Operators (0.5 hours):**
    - **Research:** Explore different types of operators:
        - **Arithmetic:** +, -, *, /, MOD (remainder), DIV (integer division).
        - **Relational/Comparison:** >, <, >=, <=, == (equals), != (not equals).
        - **Logical:** AND, OR, NOT.
    - **Pseudocode Practice:** Write pseudocode examples demonstrating the use of each type of operator.
3. **Input and Output (0.5 hours):**
    - **Research:** How do programs get data from a user (input) and display results (output)?
    - **Pseudocode Practice:** Write pseudocode to:
        - Ask the user for their name and store it in a variable.
        - Display a greeting message using the stored name (e.g., "Hello, [Name]!").
4. **Control Structures (2 hours):**
    - **Sequence:** Understand that instructions are executed one after another in order.
    - **Selection (IF/ELSE, CASE/SWITCH) (1 hour):**
        - **Research:** Learn how programs make decisions based on conditions.
        - **Pseudocode Practice:**
            - Write pseudocode to check if a user's age is 18 or over. If it is, display "You are an adult." Otherwise, display "You are a minor."
            - Write pseudocode using a CASE (or SWITCH) structure to display a different message based on a user's input of 'A', 'B', or 'C' (e.g., 'A' -> "Excellent", 'B' -> "Good", 'C' -> "Average").
    - **Iteration/Loops (FOR, WHILE) (1 hour):**
        - **Research:** Learn how programs repeat instructions.
        - **Pseudocode Practice:**
            - Write pseudocode using a FOR loop to print numbers from 1 to 5.

        - Write pseudocode using a WHILE loop to keep asking the user for a password until they enter "secret".

5. **Subroutines/Functions (0.5 hours):**
    - **Research:** Understand the concept of breaking down a large program into smaller, reusable blocks of code (subroutines, functions, procedures). Why is this useful?
    - **Pseudocode Practice:** Write pseudocode for a simple function called CalculateArea that takes length and width as inputs and returns their product.

Task 4: Putting It Together - Mini-Challenge (Approx. 1 Hour)

Objective: To apply all the learned concepts to solve a small problem using pseudocode and/or flowcharts.

Challenge: Create a pseudocode algorithm for a simple program that:

1. Asks the user to enter a temperature in Celsius.
2. Converts the Celsius temperature to Fahrenheit using the formula: $F=C\times1.8+32$.
3. Displays both the original Celsius temperature and the calculated Fahrenheit temperature.
4. Additionally, if the Fahrenheit temperature is above 80, display "It's hot!", otherwise display "It's not too hot."

Guidance:

- Think about the data types needed for temperatures.
- Consider the input and output.
- Use arithmetic operators for the conversion.
- Use a selection (IF/ELSE) statement for the temperature message.
- You can also try drawing a flowchart for this problem.

Supporting material

Here are links to relevant videos from the "Craig 'n' Dave" YouTube channel for each of the activities in the "A-Level Computer Science Pre-Course Tasks" document. Please note that some topics are covered within broader videos, and I've tried to select the most appropriate ones.

Task 1: Understanding Basic Data Types

- **What are Data Types? & Common Data Types (Integer, Real/Float, Character, String, Boolean):**
    - 72. OCR A Level (H046-H446) SLR13 – 1.4 Primitive data types (This video covers Integers, Reals/Floats, Chars, Strings, and Booleans.)
    - For a more specific look at **Strings**: AQA A'Level SLR01 Introduction to programming Part 5 – String handling
    - For **Boolean logic** (which underpins the Boolean data type): 2.4 – Boolean logic (This is a category with multiple videos; focus on the introductory ones.)
    - For **Characters**: 80. AQA GCSE (8525) SLR13 - 3.3 Characters

Task 2: The Binary Number System

- **Why Binary?**
    - CAMBRIDGE IGCSE Topic 1.1 How and why computers use binary to represent all forms of data
- **Bits and Bytes:**
    - AQA A'Level SLR10 Bits, bytes and unit representation
- **Denary to Binary Conversion:**
    - 15. OCR GCSE (J277) 1.2 Converting between denary & 8 bit binary
- **Binary to Denary Conversion:**
    - The video above for "Denary to Binary Conversion" also covers converting binary back to denary.
- **Binary Addition:**
    - 16. OCR GCSE (J277) 1.2 Adding two 8 bit binary integers
    - 76. OCR A Level (H046-H446) SLR13 – 1.4 Binary addition and subtraction
- **Introduction to Hexadecimal:**
    - Converting Denary to Hexadecimal | OCR GCSE J277

Task 3: Core Programming Concepts

- **Variables and Constants:**
    - CAMBRIDGE IGCSE Topic 8.1 Variables and constants
- **Operators (Arithmetic, Relational, Logical):**
    - 51. AQA GCSE (8525) SLR8 – 3.2 Arithmetic operators (This video covers arithmetic and comparison operators. For logical operators, refer back to the Boolean logic videos in Task 1.)
- **Input and Output:**

    - A level OCR: SLR03 - Input, output and storage (This is a playlist; look for videos specifically on input/output *concepts* rather than just hardware.)

- **Control Structures (Sequence, Selection, Iteration):**
    - **Sequence:** 63. OCR GCSE (J277) 2.2 The 3 basic programming constructs (This video introduces all three constructs.)
    - **Selection (IF/ELSE, CASE/SWITCH):** OCR J277 GCSE 2.2.1: Selection
    - **Iteration/Loops (FOR, WHILE):** GCSE Computer Science Python #5 - Iteration (while and for loops)
- **Subroutines/Functions:**
    - 48. AQA GCSE (8525) SLR8 - 3.2 Introduction to subroutines