Summer Transition Work

Task 1 – Web Development

Create a single webpage about a topic of the GCSE Computer Science course. You can choose any of the following topics-

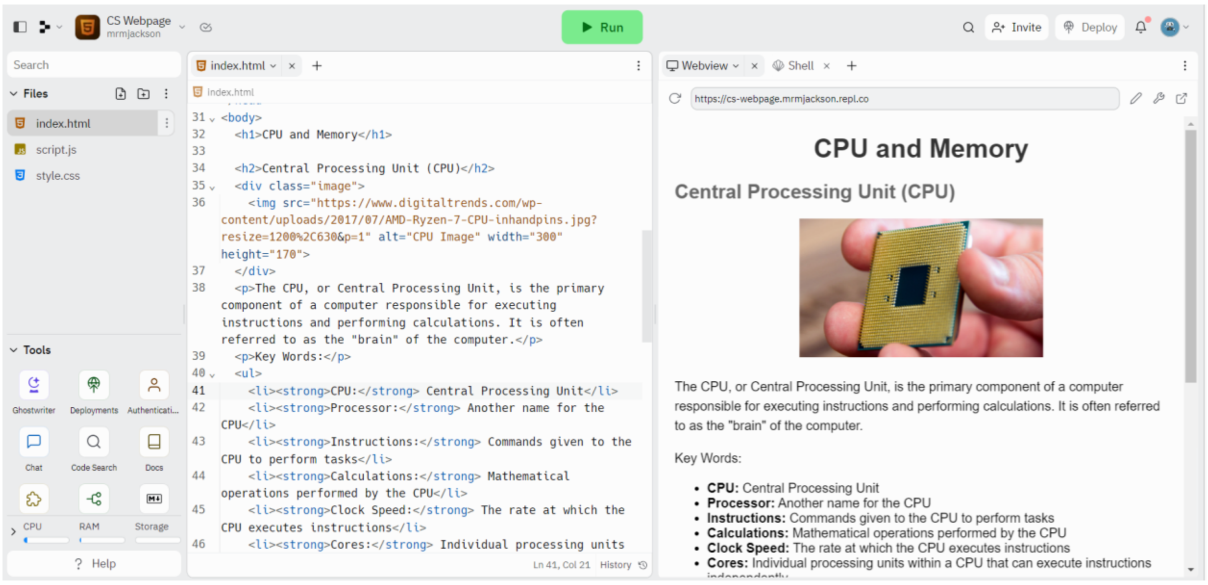
- CPU, Memory and Storage
- Networks
- Binary Numbers and Hexadecimal
- Images and Sound

Your webpage needs to include-

- Main heading and subheadings
- Information about the topic
  - Keywords and definitions
  - Explanation of the topic
  - Relevant images or diagrams
- Links to other pages or websites (Bitesize, YouTube etc)

Here is an example (it still needs some work on the colour scheme and styling)

Link - [index.html - CS Webpage - Replit]



```html
<body>
  <h1>CPU and Memory</h1>

  <h2>Central Processing Unit (CPU)</h2>
  <div class="image">
    <img src="https://www.digitaltrends.com/wp-content/uploads/2017/07/AMD-Ryzen-7-CPU-inhandpins.jpg?resize=1200%2C630&p=1" alt="CPU Image" width="300" height="170">
  </div>
  <p>The CPU, or Central Processing Unit, is the primary component of a computer responsible for executing instructions and performing calculations. It is often referred to as the "brain" of the computer.</p>
  <p>Key Words:</p>
  <ul>
    <li><strong>CPU:</strong> Central Processing Unit</li>
    <li><strong>Processor:</strong> Another name for the CPU</li>
    <li><strong>Instructions:</strong> Commands given to the CPU to perform tasks</li>
    <li><strong>Calculations:</strong> Mathematical operations performed by the CPU</li>
    <li><strong>Clock Speed:</strong> The rate at which the CPU executes instructions</li>
    <li><strong>Cores:</strong> Individual processing units
```

CPU and Memory

Central Processing Unit (CPU)

The CPU, or Central Processing Unit, is the primary component of a computer responsible for executing instructions and performing calculations. It is often referred to as the "brain" of the computer.

Key Words:

- **CPU:** Central Processing Unit
- **Processor:** Another name for the CPU
- **Instructions:** Commands given to the CPU to perform tasks
- **Calculations:** Mathematical operations performed by the CPU
- **Clock Speed:** The rate at which the CPU executes instructions
- **Cores:** Individual processing units within a CPU that can execute instructions

How to get started-

1. Create a new HTML, CSS, Javascript project in Repl
2. Edit the index.html page to include the contents of your page
3. Use **w3schools** to learn how to edit the way it looks and the layout [HTML Tutorial (w3schools.com)] This website will show you examples of loads of different HTML tags that you can use to create your webpage.
4. Link to my example page - [index.html - CS Webpage - Replit]

Task 2 – Python Challenges

Most of our programming work will be completed using Python, therefore it is vital that you programming skills are kept fresh.

Select and complete some of these challenges. Choose ones that are appropriate-

Shorter Challenges

Mathematical Operators

Challenge 1: Arithmetic Calculator

Write a program that acts as a basic arithmetic calculator. The program should take two numbers as input from the user and perform the four basic mathematical operations (addition, subtraction, multiplication, and division). The program should then display the results to the user.

[Add a screenshot of your code here]

Challenge 2: Temperature Converter

Create a program that converts temperatures between Celsius and Fahrenheit. The program should ask the user to input a temperature in either Celsius or Fahrenheit and then convert it to the other unit. Use the following formulas for conversion:
- Celsius to Fahrenheit: F = (C * 9/5) + 32
- Fahrenheit to Celsius: C = (F - 32) * 5/9

[Add a screenshot of your code here]

Challenge 3: Quadratic Equation Solver

Write a program that solves a quadratic equation. The program should prompt the user to enter the coefficients (a, b, and c) of the quadratic equation in the form of ax^2 + bx + c = 0. The program should then calculate and display the roots of the equation, handling both real and complex roots. You can use the quadratic formula to solve the equation:
- x = (-b ± √(b^2 - 4ac)) / (2a)

[Add a screenshot of your code here]

Lists

Challenge 1: Student Grades Analysis

Write a program that takes input from the user in the form of a list of student grades (ranging from 0 to 100) and performs the following tasks:

1. Calculate and output the average grade of the students.
2. Determine and output the highest and lowest grades in the list.
3. Count and output the number of students who passed (grades 40 or above) and failed (grades below 40).

Example:
Input: [75, 80, 65, 55, 90, 40, 30]
Output:
Average grade: 61.43
Highest grade: 90

Lowest grade: 30
Number of students who passed: 4
Number of students who failed: 3

[Add a screenshot of your code here]

Challenge 2: Shopping List

Write a program that allows the user to create a shopping list. The program should perform the following tasks:

1. Ask the user to enter the items they want to buy, one item at a time. The user can enter "done" when they finish.
2. Store the items in a list.
3. Output the complete shopping list.

Example:

```
Enter an item (enter "done" when finished): Apples
Enter an item (enter "done" when finished): Bread
Enter an item (enter "done" when finished): Milk
Enter an item (enter "done" when finished): done

Shopping List:
- Apples
- Bread
- Milk
```

[Add a screenshot of your code here]

Challenge 3: Number Sorting

Write a program that takes a list of numbers as input from the user and performs the following tasks:

1. Output the original list of numbers.
2. Sort the list in ascending order.
3. Output the sorted list.

Example:

```
Enter a number (enter "done" when finished): 7
Enter a number (enter "done" when finished): 3
Enter a number (enter "done" when finished): 10
Enter a number (enter "done" when finished): done

Original list: [7, 3, 10]
Sorted list: [3, 7, 10]
```

Note: To handle the user input in the challenges, you can use a while loop that prompts the user for input until a specific condition (e.g., entering "done") is met.

[Add a screenshot of your code here]

2D Lists (Harder challenges)

Use 2D lists in your solutions to these problems

Challenge 1: Matrix Transposition

Write a program that takes a 2D array as input and transposes it, swapping the rows and columns. The program should prompt the user to enter the dimensions of the matrix and its elements. The transposed matrix should then be printed as output.

[Add a screenshot of your code here]

Challenge 2: Spiral Matrix

Create a program that generates a spiral matrix of size N x N. The program should prompt the user to enter the value of N and then construct the spiral matrix. The elements of the matrix should be filled in a clockwise spiral pattern, starting from the top-left corner and moving towards the center. Finally, print the generated matrix.

[Add a screenshot of your code here]

Challenge 3: Tic-Tac-Toe Game Board

Design a program that simulates a Tic-Tac-Toe game board using a 2D array. The program should provide a user interface where two players can take turns entering their moves. The board should be displayed after each move, and the program should check for a winning condition (three in a row, column, or diagonal) or a draw. The game should continue until a player wins or the game ends in a draw.

[Add a screenshot of your code here]

String Manipulation

Challenge 1: Palindrome Checker

Write a program that checks whether a given word or phrase is a palindrome or not. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward. The program should ignore spaces and punctuation marks while checking for palindromes.

[Add a screenshot of your code here]

Challenge 2: String Reversal

Create a program that takes a string as input and reverses it. Implement the reversal logic without using any built-in string reversal functions or methods.

[Add a screenshot of your code here]

Challenge 3: Word Count

Write a program that counts the number of words in a given sentence or paragraph. Consider a word as any sequence of characters separated by whitespace. Punctuation marks should be excluded from the word count.

[Add a screenshot of your code here]

MOD and DIV

Use MOD (%) and DIV (//) in your attempts to create solutions to these problems

Challenge 1: Leap Year Checker

Write a program that prompts the user to enter a year and determines whether it is a leap year or not. A leap year is divisible by 4 but not divisible by 100, except for years that are divisible by 400.

[Add a screenshot of your code here]

Challenge 2: Palindrome Number Checker

Write a program that prompts the user to enter a number and determines whether it is a palindrome or not. A palindrome number is the same when read forward and backward. For example, 121 and 555 are palindromes.

[Add a screenshot of your code here]

Challenge 3: Factor Checker

Write a program that prompts the user to enter two numbers, A and B. Determine whether A is a factor of B or not. A is considered a factor of B if B MOD A is equal to 0.

[Add a screenshot of your code here]

Functions with Parameters

Solve these problems by creating new functions that take in parameters

Challenge 1: Calculate Area of a Rectangle

Write a function named `calculate_rectangle_area` that takes two parameters, `length` and `width`, and calculates the area of a rectangle. The function should return the calculated area.

[Add a screenshot of your code here]

Challenge 2: Find the Maximum Number

Write a function named `find_maximum` that takes three parameters, `num1`, `num2`, and `num3`, and finds and returns the maximum of the three numbers.

[Add a screenshot of your code here]

Challenge 3: Generate a Fibonacci Sequence

Write a function named `generate_fibonacci_sequence` that takes a parameter `n` and generates and returns the first `n` numbers in the Fibonacci sequence as a list. The Fibonacci sequence starts with 0 and 1, and each subsequent number is the sum of the two preceding numbers.

[Add a screenshot of your code here]

Larger Game Challenges

1. Text Adventure Game:

Create a text-based adventure game where players navigate through different scenarios by making choices. Each choice leads to different outcomes, and players can progress through the game by collecting items, solving puzzles, and interacting with characters. This game will require students to work with conditional statements, user input, and variable manipulation.

2. Word Guessing Game:

Develop a word guessing game where the computer selects a random word from a predefined list, and the player has to guess the word by entering letters one at a time. The computer provides feedback on whether the guessed letters are correct and reveals the positions of correctly guessed letters. Students will need to work with arrays, loops, string manipulation, and conditionals to create this game.

3. Quiz Game:

Design a quiz game where players are presented with a series of questions and given multiple-choice options. The player selects an answer, and the program provides feedback on whether it's correct or not. The game keeps track of the player's score and displays it at the end. Students will need to work with arrays, loops, user input, conditionals, and scoring logic to implement this game.

4. Hangman:

Description: Implement the classic game of Hangman where the player guesses letters to reveal a hidden word. The player has a limited number of attempts before the game ends.

Skills Developed: Strings, lists, loops (while or for), conditional statements, user input, and functions.