GCSE

3500U20-1

S23-3500U20-1

THURSDAY, 25 MAY 2023 – AFTERNOON

COMPUTER SCIENCE

UNIT 2: Computational Thinking and Programming

2 hours

3500U201
01

ADDITIONAL MATERIALS

You will require the electronic answer booklet (EAD) for this examination and other files for certain questions, all of which should be pre-installed on your examination account.

Your computer should be pre-installed with text editing software, a word processing package and a functional copy of the Greenfoot IDE version 2.4.2.

INSTRUCTIONS TO CANDIDATES

You will need to enter your answers to certain questions within the electronic answer document provided.

You will need to create a new plain text file to answer question 2.

You will complete the work for certain questions within the Greenfoot IDE.

Carry out all tasks and save your work regularly.

INFORMATION FOR CANDIDATES

The total number of marks available for this examination is 60.

The number of marks is given in brackets at the end of each question or part-question.

You are reminded of the need for good English and orderly, clear presentation in your answers.

1. State the effect of the following HTML tags. [5]

(a) <i>

(b) <b>

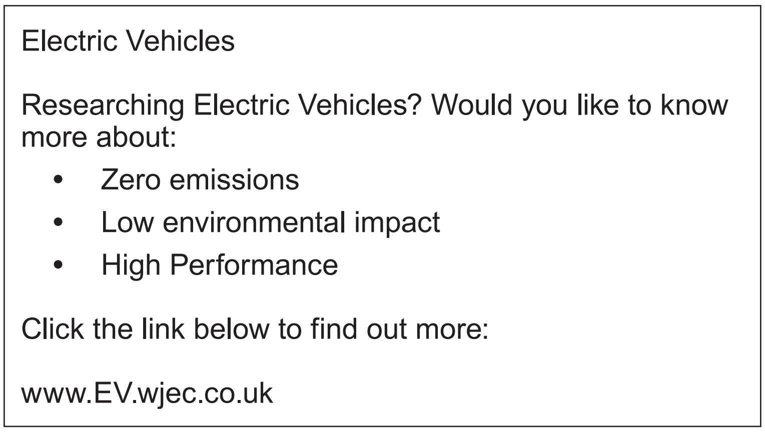
(c) <hr>

(d) <blockquote>

(e) <p>

Enter your answers in the electronic answer document.

2. A draft design for an HTML web page is shown below. [10]



Electric Vehicles

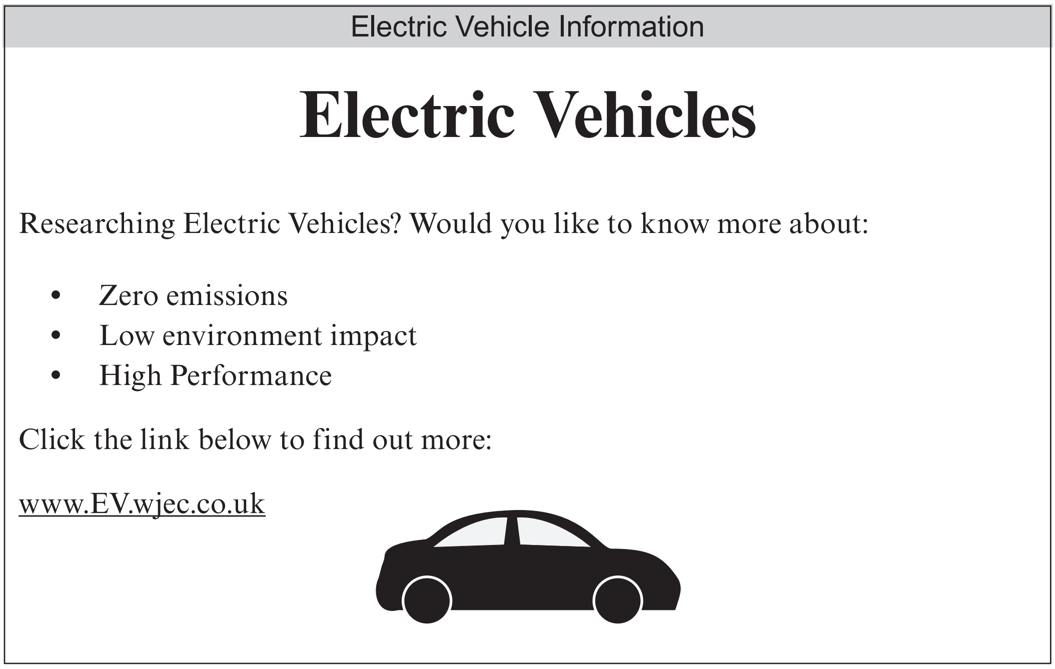
Researching Electric Vehicles? Would you like to know more about:

- Zero emissions
- Low environmental impact
- High Performance

Click the link below to find out more:

www.EV.wjec.co.uk

The design was then improved to provide the formatting and content shown.



Electric Vehicle Information

Electric Vehicles

Researching Electric Vehicles? Would you like to know more about:

- Zero emissions
- Low environment impact
- High Performance

Click the link below to find out more:

www.EV.wjec.co.uk

Copy the text from the electronic answer document into a basic text editor.

Insert the HTML tags that would be needed to display the content and formatting shown in the improved design.

The image file you require is called:	`car.jpg`
The page title should be set to:	`Electric Vehicle Information`
Save your new web page as:	`ev.txt`

3. (a) Complete the table in the electronic answer document to show all the outputs of the following assembly language program with the inputs 4 and 3: [4]

```
INP
STA first
OUT
INP
STA second
OUT
LDA first
OUT
SUB second
OUT
HLT
DAT first
DAT second
```

(b) Write an assembly language program which accepts three numerical inputs and adds them together. The program should only output the total. [6]

Enter your program into the electronic answer document.

BLANK PAGE

4. An algorithm to count the total number of cars using a section of road in 24 hours is shown. The algorithm should output the largest number of cars in an hour, the smallest number of cars in an hour and the mean number of cars per hour:

```
1    Declare Car
2      currentNumber is integer
3      maxNo is integer
4      minNo is integer
5      total is integer
6      . . .
7      set currentNumber = 0
8      set maxNo = 0
9      set minNo = 999
10     set total = 0
11     set mean = 0.0
12     for i = 1 to 24
13       . . .
14       input currentNumber
15       if currentNumber > maxNo then
16        maxNo = currentNumber
17       . . .
18       if currentNumber < minNo then
19        . . .
20       endif
21       total = total + currentNumber
22     next i
23     mean = total / 24
24     output “Total:”, total
25     output “Mean: ”, mean
26     output “Largest: ”, maxNo
27     . . .
28   . . .
```

(a) Certain lines are missing from the algorithm, indicated by . . .

Using **six** of the lines of code below, complete this algorithm in the electronic answer document. [6]

- `if currentNumber < minNo then`
- `members = TRUE`
- `output “Enter a reading for this hour:”`
- `endif`
- `mean is real`
- `minNo = currentNumber`
- `End Subroutine`
- `output “Smallest:” , minNo`

(b) Explain why the use of self-documenting identifiers is important when writing code. [3]

(c) In a certain computer there are various search algorithms available for use.

Place a cross (**X**) in the correct box in the electronic answer document to show which algorithm would be the most appropriate for sorted and unsorted data: [2]

	Sorted Data
Linear Search	☐
Diagonal Search	☐
Horizontal Search	☐
Binary Search	☐

	Unsorted Data
Linear Search	☐
Diagonal Search	☐
Horizontal Search	☐
Binary Search	☐

5. An algorithm is required to calculate the charging time (in minutes) needed for an electric vehicle to travel a certain distance (in miles).

The algorithm should:

- prompt the user to input the number of miles to be travelled
- accept the input of the number of miles to be travelled
- use the calculation:
  Time = Number of Miles * 1.5
- output the time calculated
- warn the user if the time calculated is over 60 minutes (longer than one hour)

An example of the ***input*** and output required is shown below.

```
Input number of miles: 50

Time in minutes: 75

Warning the time calculated is longer than one hour.
```

Write an algorithm to meet these requirements.
Enter your algorithm into the electronic answer document. [6]

6. A vehicle showroom would like a new scenario created in the Java programming language within the Greenfoot environment. The vehicle showroom will use the scenario as a screen saver. [5]

(a) Create a new world in the Greenfoot environment called **Advert**. Set the background image within this world to a 9 x 9 grid using the image `cell.jpg`

(b) Create a new class called **Van** and set the image of this class to `van.jpg`

(c) Populate the world with **two Van**s.

(d) Enter code into the **Van** class which will cause the **Van**s to turn and move randomly (as if driving around).

(e) Save your completed world as `finalVans`

All of the images you require are in the `Advert\images` folder.

7. Open the Greenfoot world `WJECCars7` and familiarise yourself with its contents. Complete the world as instructed below: [13]

(a) Populate the world with a single **car**, two or more **lightning** bolts and two or more **oil** drops.

(b) Edit the **lightning** and **oil** objects so that they turn and move around the world at random.

(c) Edit the **car** object so that it moves at an appropriate speed in the direction of the arrow keys when pressed.

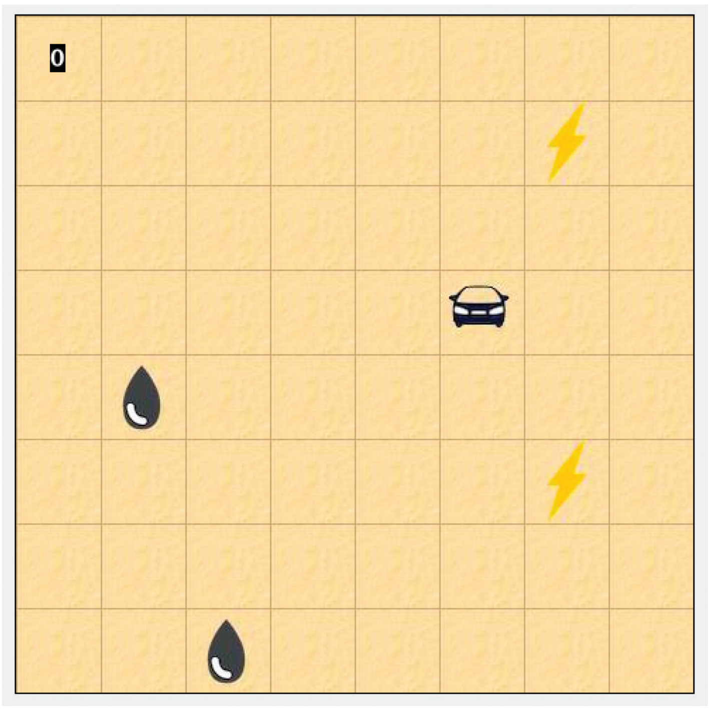
(d) Edit the **car** object so that it "collects" a **lightning** bolt when they collide (removes the **lightning** from the world).

(e) Add a sound which will play every time the **car** "collects" a **lightning** bolt.

(f) Add a **counter**. Edit the code so that the **counter** displays how many **lightning** bolts have been "collected".

(g) Edit the code so that the **counter** loses a point (1 point is deducted) if the **oil** collides with a **car**.

(h) Save your completed world as `FinalWJECCars7`



END OF PAPER

BLANK PAGE