GCSE – NEW

C500U20-1

S18-C500U20-1

COMPUTER SCIENCE – Component 2
Computational Thinking and Programming

THURSDAY, 17 MAY 2018 – AFTERNOON

2 hours

C500U201
01

ADDITIONAL MATERIALS

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

INSTRUCTIONS TO CANDIDATES

You will need to enter your answers to questions 1, 3, 4, 5, 6 and 7 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 question 8 within the Greenfoot IDE.

Carry out all tasks and make sure that you check your work carefully to ensure that the work you produce is accurate and correct.

Save your work regularly.

INFORMATION FOR CANDIDATES

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.

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

Questions 3, 5, 6, 7 and 8 will require you to draw on your knowledge from multiple areas of your course of study.

1. State the HTML tags needed for each of the following: [3]

(a) To start a new HTML document.

(b) To begin an unordered list.

(c) To create a horizontal rule.

Enter your answers in the electronic answer document.

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

Cybersecurity briefing
Staying up to date with current issues.

Hear speakers on a range of topics including:
Cyber essentials
Risk discovery
A day in the life of a hacker

All of the above in a modern, open and fascinating insight into the evolving landscape of cyber security!

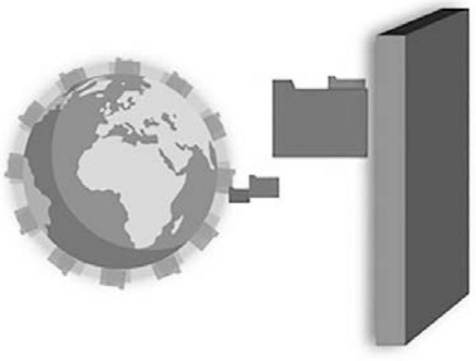
Click the link below to find out more:

www.cybersecuritybriefing.co.uk

The design was then improved using various HTML tags to provide the formatting and content shown below.

Cybersecurity briefing

Staying up to date with current issues.



Hear speakers on a range of topics including:

- Cyber essentials
- Risk discovery
- A day in the life of a hacker

All of the above in a modern, open and fascinating insight into the evolving landscape of cyber security!

Click the link below to find out more:

www.cybersecuritybriefing.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 `firewall.jpg`

The page title should be set to `Cybersecurity Briefing`

Save your new web page as `finalAdvert.txt`

3. *(a)* State the assembly language mnemonic for each of the following: [3]

(i) To load a value into a register.

(ii) To branch the operation of a program.

(iii) To end the operation of a program.

Enter your answers in the electronic answer document.

(b) Explain why it is good practice to use self-documenting identifiers in high-level code. [2]

Enter your answer in the electronic answer document.

4. Below is an algorithm:

```
multi is integer
set multi = 1
Declare TimesUp
  counter is integer
  set counter = 0
  output “Beginning a loop”
  repeat
    counter = counter + 1
    output “Counter is:” counter
    multi = multi * counter
    output “Multi is:” multi
  until counter = 3  {Note the loop has ended here}
output “Loop completed”
End Subroutine
```

Give all the outputs of the algorithm. [8]

Enter your answers in the electronic answer document.

5. An algorithm intended to monitor the production of mobile phone screens is shown below.

The algorithm requires two inputs, the number of screens produced and the size of each screen produced. If the size of the screen is too large the screen is discarded.

```
1   overLargeScreen is integer
2   requiredSize is integer
3   currentScreenSize is integer
4   numberToTest is integer
5   count is integer

6   set count = 0
7   set numberToTest = 0
8   set overLargeScreen = 0
9   set requiredSize = 101
10  set currentScreenSize  = 0

11  ...
12  input numberToTest

13  for count = 1 to numberToTest
14    output “Please enter the size of screen:” & count
15    input currentScreenSize
16   . . .
17      output “Discard this screen as it is too large.”
18      overLargeScreen = overLargeScreen +1
19     end if
20  . . .

21  output “The total number of discarded screens:” & overLargeScreen

22  . . .
```

Lines 11, 16, 20 and 22 are missing from the algorithm above.

Using four of the lines of code below, complete this algorithm in the electronic answer document. [4]

- `else is TRUE`
- `next count`
- `End Subroutine`
- `input flag`
- `if currentScreenSize < requiredSize then`
- `if currentScreenSize > requiredSize then`
- `output “Please enter the number of screens to test:”`

6. An algorithm is required to record a series of whole numbers between 1 and 65535.

The algorithm should:

- accept the input of each number
- stop accepting numbers when a rogue value is entered (a number larger than 65535)
- output the total of all the numbers entered
- output the mean of all the numbers entered
- output the largest number entered
- output the smallest number entered

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

```
Enter a number: 570

Enter a number: 1438

Enter a number: 65537


Total: 2008

Mean: 1004

Largest: 1438

Smallest: 570
```

Write an algorithm to meet these requirements. Enter your algorithm in the electronic answer document. [9]

7. Part of an algorithm intended to check the username and password entered by a user is shown below. The algorithm has two main shortcomings.

```
Declare LoginScreen
  username is string
  password is string
  counter is integer
  loggedIn is boolean
  set loggedIn = FALSE
  set counter = 0
  do
    output “Type in username”
    input username
    output “Type in password”
    input password
    if username = “User1” OR password = “Pass1” then
      output “Username and password correct”
      set loggedIn = TRUE
    else
      output “Username/password error”
      counter = counter + 1
    end if
  loop
End Subroutine
```

Explain the **two** main shortcomings with the algorithm and describe how these could be rectified. [6]

Enter your answer in the electronic answer document.

8. Open the Greenfoot world *WJECDrop8* and familiarise yourself with its contents. Complete the world as instructed below: [15]

(a) Populate the world with an **umbrella**, **sun** and at least two instances of **waterDrop**.

(b) Edit the **waterDrop** and **sun** objects so that they turn and move at random.

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

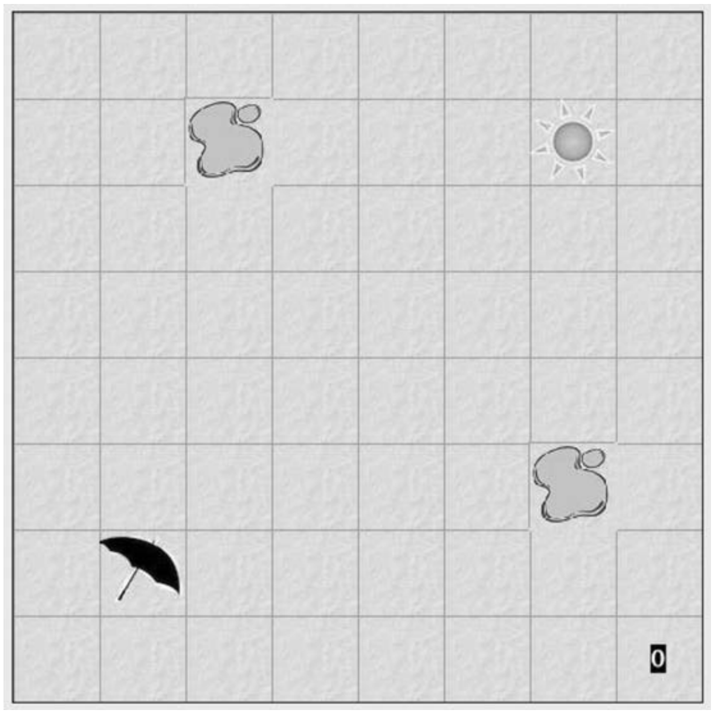
(d) Edit the **umbrella** object so that it "catches" a **waterDrop** when they collide (removes the **waterDrop** from the world).

(e) Add a sound which will play every time the **umbrella** "catches" a **waterDrop**.

(f) Add a **counter**. Edit the code so that the **counter** displays how many **waterDrops** have been "caught".

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

(h) Save your completed world as *finalDrop8*.



END OF PAPER