**Staffordshire University  
Department of Computing**

**COMP40004: Web Development & Operating Systems  
Week 6: Functions, Randoms, String Manipulation and Debugging**

Please work on these tasks individually.

# Individual Task 1: Say Favourites

Write a bash script called **say-favourites.sh** which;

1. Has a function which accepts two arguments – a favourite colour and a favourite animal
2. When called, the function should output “Your favourite colour is: {colour} and your favourite animal is: {animal}” – e.g. “Your favourite colour is: orange and your favourite animal is: cat”
3. Call the function twice, each time with a different colour and animal.

## Expected Output

When you run your script, you should see the following;

Text

Description automatically generated

## Your Answer:

Paste in the contents of say-favourites.sh below.

#!/bin/bash

# Individual Task 2: Sum Three

Write a bash script called **sum-three.sh** which;

1. Has a function, which accepts three arguments. The function should add together all of the numbers given as arguments, and uses a global variable to return the sum total of the numbers. (e.g. if given the arguments 10, 12 and 4, would return the value 26).
2. Prompts the user to enter three numbers (with three **read** statements)
3. Calls the function and provides the three numbers the user has typed
4. Outputs “The total of {num1}, {num2}, and {num3} is: {total}” – e.g. “The total of 10, 12, and 4 is 26.”
5. You should not have an **echo** statement inside of your function.

## Expected Output

When you run your script, you should see something like the following;

Text

Description automatically generated

## Your Answer:

Paste in the contents of sum-three.sh below.

Replace this text with your code.

Paste it in or include a screenshot.

# Individual Task 3: Compare Towns

Write a bash script called **towns.sh** which;

* Prompts the user to enter the town they are from (with **read**)
* Prompts the user to enter the town they live in (with **read**)
* Includes a function which takes two arguments – the town a user is from and the one they live in.
  + The function should echo both towns  
    e.g. “You are from: *hometown* and you live in: *currenttown*.”
  + The function should echo a message if the two towns are the same  
    e.g. “You live in the same town you are from”
  + The function should echo a message with the number of characters in the user’s town that they live in  
    e.g. “The town you live in has *8* letters in its name”

## Expected Output

When you run your script, you should see something like the following:

Text

Description automatically generated

## Your Answer:

Paste in the contents of towns.sh below.

Replace this text with your code.

Paste it in or include a screenshot.

# Individual Task 4: Web Hosting URL

Write a bash script called **webdrive.sh** which;

* Prompts the user to enter their student email address (with **read**)  
  e.g. [a000001a@student.staffs.ac.uk](mailto:a000001a@student.staffs.ac.uk)
* Uses a function called email\_to\_hosting\_url to:
  + Extracts the student username from this URL
  + Outputs a URL the user can use to access their student web hosting:  
    e.g. <http://web.fces.staffs.ac.uk/a000001a>
  + (Note you won't be able to access your web hosting off campus, without a VPN, and there's no need to - we're just looking to see you can do a string transformation!)
* Tips and Notes:
  + You can extract the username from the email address with an assumption that usernames are a set length
  + The student web hosting URL is in the format above, ending with the student username
  + You won’t actually be able to access the student web hosting URL off site unless you’re using the university VPN. But there’s no need to – just outputting the URL is enough for this
  + Echo the hosting URL inside the function, or return it with a global variable.

## Expected Output

When you run your script, you should see something like the following:

Graphical user interface, text

Description automatically generated

## Your Answer:

Paste in the contents of webdrive.sh below.

Replace this text with your code.

Paste it in or include a screenshot.

# Practice Task 5: Fruit Loops

Write a bash script called **fruitloops.sh** which;

* Uses an **until** loop to continuously ask the user to enter a 6 letter fruit until they enter a word with 6 letters
* Uses a **for** loop to output the user’s chosen fruit back to them 10 times.
* Uses a **while** loop to output each character of the user’s chosen fruit back to them individually.

## Expected Output

When you run your script, you should see something like the following:

Graphical user interface, text

Description automatically generated

## Your Answer:

Paste in the contents of fruitloops.sh below.

Replace this text with your code.

Paste it in or include a screenshot.

# Individual Task 6: A Buggy Script

Write a bash script which called **buggy.sh** which;

* Contains the code from Lecture B slide 20

Then;

* Call the script twice, each time redirecting (and **appending**) **stdout** to a file called output.log, and **stderr** to a file called error.log
* Verify using **cat** that you can see the successful output twice in output.log, and the errors twice in error.log

## Expected Output (and solution for what commands to enter)

When you run your script, you should see something like the following:

Text

Description automatically generated

# Individual Task 7: Number Guessing Game

Write a bash script called **numbergame.sh** which;

* Generates a random number between 0 and 20
* Uses an **until** loop to ask the user to enter a guess
  + Keeps track of the number of guesses the user enters
  + If the user’s guess is equal to the random number, stop the loop and output a congratulations message with the number of guesses the user needed.
  + If the user’s guess is not right, output either “higher” (if their guess was too high) or “lower” (if their guess was too low) and ask the user to guess again.

## Tips:

* Go step by step. Get a random number generated and output it first. Then get the guessing logic working. Only then, add the logic to count the user’s number of guesses.
* You can use the random\_to function from the lectures
* You'll need an **until** loop
* You'll need to use more than one **if** statement (or one with an **elif**) to compare whether a number is greater than, and to compare if it is less than, a given number.

## Expected Output

When you run your script, you should see something like the following:

Text

Description automatically generated

## Your Answer:

Paste in the contents of numbergame.sh below.

Replace this text with your code.

Paste it in or include a screenshot.