**Unit 3 Algorithmics**

**Submit Task – Week 2**

Warm Up

Write a function which finds the largest power of 2 which goes into a number. Do this using a loop rather than logarithms. Remember that we use \*\* to raise a number to a power in Python, e.g 2 \*\* 3 🡪 8

Output should be:

print(largestPower(8)) 🡪 3

print(largestPower(17)) 🡪 4

Next write a function which finds the number of digits in a number. You can use some of the functions listed below (e.g. logarithms), and there are multiple possible approaches.

Output should be:

print(numDigits(8)) 🡪 1

print(numDigits(100)) 🡪 3

You can use these functions in the task below.

Main Task

You will write functions to convert between decimal and binary representation in Python.

A binary number is made up of 1s and 0s. Each bit/digit corresponds to a power of 2, starting at zero. For example:

1001 = **1** x 23 + **0** x 22 + **0** x 21 + **1** x 20 = 8 + 1 = 9.

10010 = **1** x 24 + **0** x 23 + **0** x 22 + **1** x 21 + **0** x 20 = 16 + 2 = 18.

First write a function to convert from binary to decimal. Work through each bit of the binary representation and raise two to its power, adding the result each time.

Check that your function gives the following answers.

print(toDecimal(1001)) 🡪 9

print(toDecimal(1000)) 🡪 7

print(toDecimal(10101)) 🡪 21

print(toDecimal(11000)) 🡪 24

Converting from decimal to binary is a bit harder. First consider the mathematics.

e.g. 27

Find the largest power of 2 that is smaller than or equal to your number.

27 > 16, 27 < 32. 16 = 24 so 27 will have five digits in binary.

The left most digit is worth 24 = 16.

27 ≥ 16 so set the first digit to 1.

Now reduce 27 by 16, leaving us with 11.

Consider the next power down, i.e. 8.

11 ≥ 8 so set the second digit to 1.

Reduce 11 by 8, leaving us with 3.

3 < 4 so set the third digits to 0.

No need to reduce here.

3 ≥ 2 so set the fourth digits to 1.

Reduce 3 by 2, leaving us with 1.

1 ≥ 1 so set the fifth digit to 1.

We’ve reached the last power so stop.

Result: 11011.

Check that your function gives you the following answers:

print(toBinary(27)) 🡪 11011

print(toBinary(256)) 🡪 100000000 (8 zeros)

print(toBinary(170)) 🡪 10101010

print(toBinary(65535)) 🡪 1111111111111111 (16 ones)

Functions you might need

int(‘899’) – converts the string ‘899’ to an integer 899

str(899) – converts the integer 899 to a string ‘899’.

If you want to use logarithms, put the line import numpy as np at the top of your program. (If using VS Code, you’ll need to run pip install numpy from the terminal.) Then you can use:

np.log10(1020) – returns the log of 1020 in base 10

np.floor(10.3) – returns the whole number part of 10.3, i.e. 10.

np.ceil(10.3) – returns the number rounded up, i.e. 11.

If you struggle…

That’s good! I would rather see your half-written program which doesn’t quite work, than something you’ve copied from the internet or asked ChatGPT to do for you. Getting stuck and problem solving is an important part of programming. This in turn will help you to test your algorithms for accuracy.