Lab 1.1 (Deadline Friday 21 January 23:59)

- Upload your code to Einstein to have it verified.

Chop

- Write a Python program called `chop_011.py` that reads lines of text from `stdin`. Each line consists of a single string. The program should print out the string minus its first and last characters. If there is nothing left after removing the first and last characters the program should not print anything. For example:

```
$ cat chop_stdin_00_011.txt
Jimmy
ran
to
a
standstill
```

```
$ python3 chop_011.py < chop_stdin_00_011.txt
imm
a
tandstil
```

Capitals

- Write a program called `capitals_011.py` that reads lines of text from `stdin`. Each line consists of a single string. The program should capitalise the first and last characters of the string and print the result. If the string has fewer than two characters the program should print nothing. For example:

```
$ cat capitals_stdin_00_011.txt
mittens
mit
m
pi
pittance
```

```
$ python3 capitals_011.py < capitals_stdin_00_011.txt
MittenS
MiT
PI
PittancE
```

Middle

- Write a program called `middle_011.py` that reads lines of text from `stdin`. Each line consists of a single string. The program should print out the middle character of each string. If the string does not have a middle character the program should print “No middle character!”. For example:

```
$ cat middle_stdin_00_011.txt
marshmallow
pip
p
pips
zingy
```

```
$ python3 middle_011.py < middle_stdin_00_011.txt
m
i
p
No middle character!
n
```

Substring

- Write a program called *`substring_011.py`* that reads lines of text from `stdin`. Each line consists of two strings. The program should print `True` if the first string is a substring of the second (and `False` otherwise). Note that differences in case can be ignored. For example:

```
$ cat substring_stdin_00_011.txt
Rump Rumpelstiltskin
rump Rumpelstiltskin
stilt Rumpelstiltskin
stiLT Rumpelstiltskin
pest Rumpelstiltskin
up Rumpelstiltskin
```

```
$ python3 substring_011.py < substring_stdin_00_011.txt
True
True
True
True
False
False
```

Contains

- Write a program called *`contains_011.py`* that reads lines of text from `stdin`. Each line consists of two strings. The program should print `True` if each of the characters in the first string is also in the second string (and `False` otherwise). Note that once a character has been matched in the second string it cannot be matched again. Also note that differences in case can be ignored. For example:

```
$ cat contains_stdin_00_011.txt
c cat
AC cat
tac caT
ttac cat
```

```
$ python3 contains_011.py < contains_stdin_00_011.txt
True
True
True
False
```

- Hint: You may find the `str.replace()` method useful. Use `help` or `pydoc` to look it up.

Emails

- Write a program called *`emails_011.py`* that reads DCU student email addresses from `stdin`. For each email address the program should print out the corresponding student's name. For example:

```
$ cat emails_stdin_00_011.txt
valerie.maguire2@mail.dcu.ie
fred.quinn33@mail.dcu.ie
jimmy.clancy5@mail.dcu.ie
```

```
$ python3 emails_011.py < emails_stdin_00_011.txt
Valerie Maguire
Fred Quinn
Jimmy Clancy
```

First M

- Write a program called *`firstm_011.py`* that reads lines of text from `stdin`. The program should print each line with the first word that begins with a lower case `m` now capitalized. For example:

```
$ cat firstm_stdin_00_011.txt
Mickey Mouse was a kind of mouse.
Mickey Mouse was a kind of mouse. A mouse with a sense of humour.
```

```
$ python3 firstm_011.py < firstm_stdin_00_011.txt
Mickey Mouse was a kind of Mouse.
Mickey Mouse was a kind of Mouse. A mouse with a sense of humour.
```

Water

- You have some water and some buckets to fill.
- Write a program called *`water_011.py`* that reads two lines of text from `stdin`.
- Line 1 contains a single integer, N, the number of litres of water available. N is in the range 0-1000.
- Line 2 lists the capacity in litres of one or more buckets. The capacity of each bucket is specified by a positive integer.

- Buckets must be filled in the order specified on line 2.
- Your program should output the number of buckets that can be completely filled before you run out of water.
- In this example we have 10 litres of water. We fill the first bucket (taking 6 litres), we fill the second bucket (taking another 2 litres) but we run out of water before we have completely filled the third bucket (it requires 5 litres). We output 2 (the number of buckets completely filled):

```
$ cat water_stdin_00_011.txt
10
6 2 5 1 1
```

```
$ python3 water_011.py < water_stdin_00_011.txt
2
```