Lab 2.1 (Deadline Friday 28 January 23:59)

- Upload your code to Einstein to have it verified.

Anagrams

- Two words are anagrams if the letters of one word can be rearranged to form the other word. For example *angel* and *glean* are anagrams. Write a Python program called `anagram_021.py` that reads in pairs of words (one pair per line) from `stdin` and prints `True` if the pair are anagrams and `False` otherwise. For example:

```
$ cat anagram_stdin_00_021.txt
cinema iceman
dog god
house car
stub buts
angel glean
aangl angel
a aardvark
aardvark a
```

```
$ python3 anagram_021.py < anagram_stdin_00_021.txt
True
True
False
True
True
False
False
False
```

Palindromes

- A palindrome is a word, phrase, number or other sequence of characters which reads the same backwards as forwards. Allowances are made for capital letters, punctuation and white space (word dividers). Write a program called `palindrome_021.py` that reads lines of text from `stdin` and prints `True` if the line is a palindrome and `False` otherwise. For example:

```
$ cat palindrome_stdin_00_021.txt
Step on no pets.
Step on no cats.
Able was I ere I saw Elba.
Doc, note: I dissent. A fast never prevents a fatness. I diet on cod.
Bananas
Abel was I ere
A
Aa
ABa
123.
1221.
4444444.
This is not a palindrome.
```

```
$ python3 palindrome_021.py < palindrome_stdin_00_021.txt
True
False
True
True
False
False
True
True
True
False
True
True
False
```

- Hints:

    1. Convert the string to lowercase first. Use `pydoc` to check out the `str` class documentation. You need to find a method that will do the conversion for you.
    2. You need to strip out any characters which are not alphanumeric. Again, use `pydoc` to check out the `str` class documentation to find methods that will help you.
    3. Once you have the string in canonical form (i.e. in lowercase with all non-alphanumeric characters removed) then simply check whether it is the same sequence backwards as forwards.

Unique word count

- Write a program called *`unique_021.py`* that counts the total number of unique words in lines of text read from `stdin`. Running the program against gettysburg.txt should produce the following output:

```
$ python3 unique_021.py < gettysburg.txt
143
```

- Hints:

    1. You will have to remove *surrounding* punctuation in the text. For example *house* and *house.* should not be counted as separate unique words. For this task you may find `string.punctuation` useful.
    2. You will have to cater for upper and lower case versions of words. For example *Four* and *four* should not be counted as separate unique words.
    3. Only alphanumeric tokens are to be counted as words. For example *November* and *19* are words but – is not.
    4. Here is the sorted list of what my code considers a unique word:

```
['1863', '19', 'a', 'above', 'abraham', 'add', 'advanced', 'ago', 'all', 'altoge
```

Birthday

- Write a program called *`birthday_021.py`* that reads lines of text from `stdin` where each line consists of a person's date of birth. A date of birth is specified by three integers: a day, a month and a year. The program should determine on which day of the week each person was born and print the corresponding line from the poem:

  Monday's child is fair of face
  Tuesday's child is full of grace
  Wednesday's child is full of woe
  Thursday's child has far to go
  Friday's child is loving and giving
  Saturday's child works hard for a living
  Sunday's child is fair and wise and good in every way

  For example:

```
$ cat birthday_stdin_00_021.txt
1 3 1990
12 10 1992
9 5 1995
```

```
$ python3 birthday_021.py < birthday_stdin_00_021.txt
You were born on a Thursday and Thursday's child has far to go.
You were born on a Monday and Monday's child is fair of face.
You were born on a Tuesday and Tuesday's child is full of grace.
```

- Hint: Import the `calendar` module. Use `pydoc` to look up the `calendar.weekday()` function. It will do the hard work for you.

ABC

- Write a program called *`abc_021.py`* that reads two lines of text from `stdin`.
- The first line consists of three numbers: A, B, C. The numbers can be in any order but we know that A < B < C.
- The second line specifies the order that your program should output the numbers. For example:

```
$ cat abc_stdin_00_021.txt
6 4 2
CAB
```

```
$ python3 abc_021.py < abc_stdin_00_021.txt
6 2 4
```

- Here is another example:

```
$ cat abc_stdin_01_021.txt
1 5 3
```

```
ABC
```

```
$ python3 abc_021.py < abc_stdin_01_021.txt
1 3 5
```