Lab 5

COMP9021, Session 2, 2015

1 ☞ Magic squares

A 3×3 square whose cells contain every digit in the range 1–9 is said to be *magic* if the sums of the rows, the sums of the columns and the sums of the diagonals are all equal numbers.

Write a program that generates all magic squares.

Here is the beginning of a possible run of the program ("possible", as the order of the output solutions can vary):

```
$ python3 question_1.py
  4  9  2
  3  5  7
  8  1  6

  6  7  2
  1  5  9
  8  3  4

  ...
```

2 ☞ Extracting information from a web page

Write a program that extracts titles from a front page of the Sydney Morning Herald, provided under the name SMH.txt, meant to be saved in the working directory. You are provided with the expected output, saved in the file `question_2_outputs.txt`, though you might do a better job and remove some of the titles (for instance, *The Lady who lives on the Moon* could go...). Make sure that the output does not include any unwanted HTML entity.

For this question, you probably need to use regular expressions.

3 ☞ A calendar program

Write a program that provides a variant on the Unix `cal` utility (in particular because it lets the weeks start on Monday, not Sunday), following this kind of interaction:

```
$ python3 calendar.py
I will display a calendar, either for a year or for a month in a year.
The earliest year should be 1753.
For the month, input at least the first three letters of the month's name.
Input year, or year and month, or month and year: 3194
                               3194

       January               February               March
 Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
                 1  2      1  2  3  4  5  6      1  2  3  4  5  6
  3  4  5  6  7  8  9   7  8  9 10 11 12 13   7  8  9 10 11 12 13
 10 11 12 13 14 15 16  14 15 16 17 18 19 20  14 15 16 17 18 19 20
 17 18 19 20 21 22 23  21 22 23 24 25 26 27  21 22 23 24 25 26 27
 24 25 26 27 28 29 30  28                    28 29 30 31
 31
        April                  May                   June
 Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
              1  2  3                     1         1  2  3  4  5
  4  5  6  7  8  9 10   2  3  4  5  6  7  8   6  7  8  9 10 11 12
 11 12 13 14 15 16 17   9 10 11 12 13 14 15  13 14 15 16 17 18 19
 18 19 20 21 22 23 24  16 17 18 19 20 21 22  20 21 22 23 24 25 26
 25 26 27 28 29 30     23 24 25 26 27 28 29  27 28 29 30
                       30 31
         July                 August              September
 Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
              1  2  3   1  2  3  4  5  6  7            1  2  3  4
  4  5  6  7  8  9 10   8  9 10 11 12 13 14   5  6  7  8  9 10 11
 11 12 13 14 15 16 17  15 16 17 18 19 20 21  12 13 14 15 16 17 18
 18 19 20 21 22 23 24  22 23 24 25 26 27 28  19 20 21 22 23 24 25
 25 26 27 28 29 30 31  29 30 31              26 27 28 29 30
       October               November              December
 Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
                 1  2      1  2  3  4  5  6            1  2  3  4
  3  4  5  6  7  8  9   7  8  9 10 11 12 13   5  6  7  8  9 10 11
 10 11 12 13 14 15 16  14 15 16 17 18 19 20  12 13 14 15 16 17 18
 17 18 19 20 21 22 23  21 22 23 24 25 26 27  19 20 21 22 23 24 25
 24 25 26 27 28 29 30  28 29 30              26 27 28 29 30 31
 31
```

In doing this exercise, you will have to find out (or just remember. . .) how leap years are determined, and what is so special about the year 1753. . .

```
$ python3 calendar.py
I will display a calendar, either for a year or for a month in a year.
The earliest year should be 1753.
For the month, input at least the first three letters of the month's name.
Input year, or year and month, or month and year: 3194 Sept
    September 3194
 Mo Tu We Th Fr Sa Su
              1  2  3  4
  5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30
$ python3 calendar.py
I will display a calendar, either for a year or for a month in a year.
The earliest year should be 1753.
For the month, input at least the first three letters of the month's name.
Input year, or year and month, or month and year: dEcEm 3194
    December 3194
 Mo Tu We Th Fr Sa Su
              1  2  3  4
  5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30 31
```

4 Sierpinski triangle

Write a program that generates Latex code, a `.tex` file, that can be processed with pdflatex to create a `.pdf` file that depicts Sierpinski triangle, obtained from Pascal triangle by drawing a black square when the corresponding number is odd. A simple method is to use a particular case of Luca's theorem, which states that the number of ways of choosing k objects out of n is odd iff all digits in the binary representation of k are digits in the binary representation of n. For instance:

- $\binom{5}{3} = 10$, which corresponds to a white square as 10 is even; indeed, 5 is 101 in binary, 3 is 11 in binary, and there is at least one bit set to 1 in 11 (namely, the leftmost one), which is not set to 1 in 101;
- $\binom{6}{2} = 15$, which corresponds to a black square as 15 is odd; indeed, 6 is 110 in binary, 2 is 10 in binary, and all bits (actually, the only bit) set to 1 in 10 are set to 1 in 110.

So your program has to generate a file named `Sierpinski_triangle.tex`, similar to the one provided; examine the contents of this file to see which text needs to be output.

The file `Sierpinski_triangle.pdf` is also provided, but if you want to generate it yourself from `Sierpinski_triangle.tex`, you need to have Tex installed on your computer (install it if that is not the case, see `http://www.tug.org/texlive/`), and then execute

```
pdflatex Sierpinski_triangle.tex
```

from the command line, or open `Sierpinski_triangle.tex` in the Latex editor that comes with your distribution of Tex, and it will just be a matter of clicking a button...