Exercise no. 1:

A «leap year» (Schaltjahr) is

- a year exactly divisible by 4, but not by 100

or

- a year exactly divisible by 400

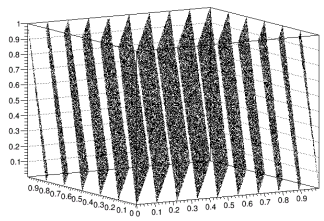
Write a program which asks the user the year as input and says whether it is a leap year or not. The program should check whether the inserted year is <= 0 and ask the user to insert the year again until a year > 0 is given.

Exercise no. 2:

RANDU is a simple Linear Congruential Generator distributed by IBM in 1960.
It generates random numbers following the algorithm:

$$x_i = 65539 x_{i-1} \bmod 2^{31}$$

It was widely used until some defects that make it unsuitable for many Monte Carlo problems was discovered.
One of those defects is that if it is used to generate 3D points (x,y,z) over the unit cube $[0,1]^3$ the points don't appear to be uniformly distributed but they rather lie in 15 hyperplanes, as shown in the fig.



Your tasks are:

- verify this defect of the generator
- store generated (x,y,z) points in a text file

Write a program to implement the RANDU generator setting the initial seed to 1. Then generate a sequence of 300000 random numbers and scale them to the interval unit (0,1) in order to obtain 100000 (x,y,z) points.
First three generated numbers will be used to generate the first point (x_1,y_1,z_1) and so on. For any point (x_i,y_i,z_i) it comes out that the quantity $9x_i - 6y_i + z_i$ is always integer with values restricted to an interval [N_{min},N_{max}]. Find the values N_{min}, N_{max} and print them to the screen.
Fill in a data.txt file with the point coordinates as:

x_1 y_1 z_1
x_2 y_2 z_2
...
x_{100000} y_{100000} z_{100000}

nb: if you want to verify the correctness of your implementation of the RANDU generator you can use Root executing the two commands:

```
> TGraph2D g("data.txt");
> g.Draw("P*");
```

Exercise no. 3:

The Runge-Kutta-Fehlberg method is an adaptive stepsize method to solve ordinary differential equations. The stepsize is indeed not fixed but is decided at each step.

To advance the solution from (x_0,y_0) to (x_1,y_1) the method starts from an initial stepsize h and calculates:

$$k_1 = f(x_0, y_0)$$

$$k_2 = f(x_0 + h, y_0 + hk_1)$$

$$k_3 = f(x_0 + \tfrac{1}{2}h, y_0 + h\tfrac{k_1+k_2}{4})$$

$$y_1^A = y_0 + \tfrac{h}{2}(k1 + k2)$$

$$y_1^B = y_0 + \tfrac{h}{6}(k_1 + k_2 + 4k_3)$$

and then evaluates the error as $|error| \approx \left|y_1^B - y_1^A\right|$

If the error is smaller than a given tolerance T the stepsize h is accepted and the solution advances to the point

$$x_1 = x_0 + h$$

$$y_1 = y_1^B$$

Otherwise (if the error is greater than T) the stepsize is rejected and a smaller stepsize is defined as:

$$h_{new} = 0.9h\left(\frac{T}{\left|y_1^B - y_1^A\right|}\right)^{1/3}$$

The process is iterated until the error becomes smaller than T.

Use this method to solve the differential equation $y' = y - x$
with initial condition $y(0) = 2/3$
setting h = 0.1 as initial stepsize (at each step) and T=0.0001
Advance the solution up to the value x = 3 and print the solution y(3) to the screen. **Please observe that at the last step you need to hit *x* = 3 exactly, so you will need to shrink the last value of h accordingly.**

N.B: The true value is *y*(3) = -2.69518.... but remember that $\left|y^B(3) - y^A(3)\right| < T$ does not necessarily implies that $\left|y^B(3) - y^{true}(3)\right| < T$)