PROGRAMACIÓN LINEAL
PROYECTO I

1. Implementa la Fase II **revisada** para un problema del tipo

$$\begin{aligned} &\min && \boldsymbol{c}^\top \boldsymbol{x} \\ &\text{sujeto a} && A\boldsymbol{x} \leq \boldsymbol{b} \\ & && \boldsymbol{b},\, \boldsymbol{x} \geq \boldsymbol{0}\,. \end{aligned} \tag{1}$$

Antes de entrar al ciclo de Fase II, la función debe introducir variables de holgura.
Es importante mantener los conjuntos de índices básicas y no básicas B y N.

```
function [xo, zo, ban, iter] = mSimplexFaseII(A, b, c)
% purpose: Versión del Simplex en la Fase II
%   minimizar   c^T x
%   sujeto a    Ax <= b ,   x >= 0 ,   b >= 0
%
% In : A ... mxn matrix
%      b ... column vector with as many rows as A
%      c ... column vector with as many entries as one row of A
%
% Out: xo   ... SFB óptima del problema
%      zo   ... valor óptimo del problema
%      ban  ... indica casos:
%              -1  ... si el conjunto factible es vacio
%               0  ... si se encontro una solución óptima
%               1  ... si la función objectivo no es acotada.
%      iter ... es el numeró de iteraciones (cambios de variables basicas)
%                que hizo el método
%


   ...
end
```

Tarea: Verifiquen su código con (al menos) un ejemplo que han resuelto exactamente.

Sugerencias: Un código eficiente no contiene más que 120 líneas.

2. El ejemplo de Klee–Minty (simplificado, tomado de Kitahara, Mizuno).
La Fase II revisada realiza $(2^m - 1)$ iteraciones para resolver el siguiente problema:

$$\begin{aligned} \text{minimizar} \quad & -\textstyle\sum_{i=1}^{m} x_i \\ \text{sujeto a} \quad & x_1 \leq 1 \\ & 2\textstyle\sum_{j=1}^{i-1} x_j + x_i \leq 2^i - 1\,,\ i = 2, \ldots, m \\ & x_i \geq 0\,,\ i = 1, \ldots, m\,. \end{aligned}$$

Resolver los casos $m = 3, 4, 5, 6, 7, 8, 9, 10$, y reportar el número de iteraciones en cada caso junto con el tiempo de máquina (en MatLab el tiempo se mide con: `tic` .. code .. `toc`).

m	número de iteraciones	cpu time

Sugerencias:

- Escribe una función `generaKleeMinty.m`.
- Después, llama esa y la función del apartado 1 en un script `SimplexKleeMinty.m` en el cual se genera la tabla.
- Te podrían servir los comandos: `ones, eye, tril`.

3. Estudio empírico de la complejidad computacional del método Simplex con problemas aleatorios.

Las siguientes líneas (de MatLab / Octave) generan un problema a cual se puede aplicar su método

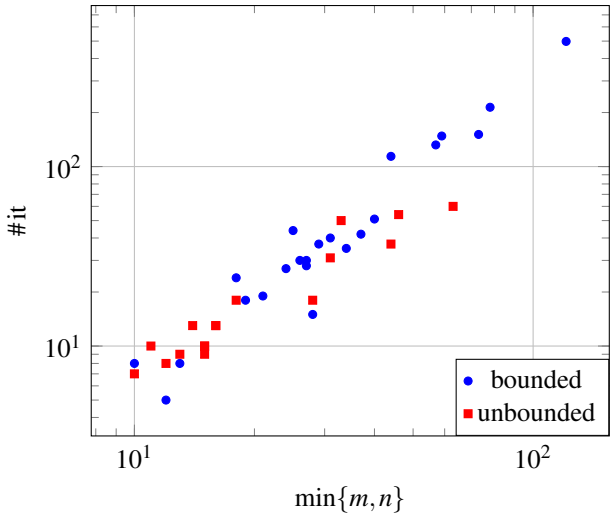
```
% generar dimensiones del problema
m = round(10*exp(log(20)*rand()));
n = round(10*exp(log(20)*rand()));

% generar A, b, c
sigma = 100;
A = round(sigma*randn(m,n));
b = round(sigma*abs(randn(m,1)));
c = round(sigma*randn(n,1));
```

Tareas:

a) correr su implementación del Simplex a (al menos) 50 problemas (generadas con las líneas arriba) y en cada caso guardar:
   - Los valores de n y m
   - El número de iteraciones que hizo su método
   - Una variable (lógica) que indique si se encontró solución óptima o si el problema no es acotado inferiormente.

b) Graficar en el eje X la dimensión $\min\{m,n\}$ y en el eje Y el número de iteraciones para cada caso. Si se encontró una solución óptima, poner una marca en azul. Si no fue acotado, poner una marca en rojo. En la gráfica use para ambos ejes la escala log. Un ejemplo de 20 casos esta en la siguiente gráfica:



Una gráfica similar se puede crear en Matlab/Octave. Unos ajustes son:

```
scatter( minmn(J_bounded), iter(J_bounded), 'b', 'filled')
hold on
scatter( minmn(J_notbounded), iter(J_notbounded), 'r', 's', 'filled')
hold off

xlabel('min(m,n)', 'fontsize', 14);
ylabel('#it', 'fontsize', 14);
set(gca,'xscale','log')
set(gca,'yscale','log')
set(gca,'YMinorTick','on')
set(gca,'XMinorTick','on')
grid on
```

c) Interprete la gráfica y deduce una expresión para la complejidad del método Simplex en el *average case*.
Sugerencia: Si $\#it = O(\min(m,n)^p)$, entonces $\log(\#it) = p \cdot \log(m+n) + C$, es decir, polinomios son rectas en un log-log-plot.

Condiciones para entregar el proyecto

- Equipos entre dos y tres integrantes.
- Fecha: Domingo 21 de Octubre a las 23:55 en comunidad itam.
- Entregar todo lo siguiente en un archivo `zip`.
  - Entregar programas y archivos:
    - `mSimplexFaseII.m` y `generaKleeMinty.m`.
    - `testFaseII.m` que verifica con un ejemplo exacto la implementación de `mSimplexFaseII.m`.
    - `SimplexKleeMinty.m` que genera la tabla usando la función `generaKleeMinty.m`.
    - a) algún *script* que muestre el resultado. En la documentación hay que describir el problema e interpretar los resultados.
      b) `SimplexEmpirico.m` que grafique el experimento de los 50 (o más) problemas aleatorios.
  - Entregar una documentación en PDF que contiene
    - nombres y claves únicas de los integrantes de equipo.
    - la interpretación de los resultados.

Nota: En el sentido de *paired programming*, les recomiendo que a lo más dos personas trabajan en un archivo. Esa debe ayudarles terminar el proyecto más rápido. Así, 1 o 2 personas pueden dedicarse a algo, mientras el resto del equipo avanza en otra parte. Eso es común en equipos de "*agile development*".