Frank Steven Hoyos Sanzhez
1007544271
1.

```
%  variables
var int: x1; %  conejos
var int: x2; %  pollos

% Función objetivo (beneficio)
var int: beneficio = 500*x1 + 300*x2;

% Restricciones
constraint 20*x1 + 10*x2 <= 1000; % Disponibilidad de pienso
constraint 3*x1 + 2*x2 <= 180; % Disponibilidad de horas
% Restricciones de no negatividad
constraint x1 >= 0;
constraint x2 >= 0;

%  maximizar
solve maximize beneficio;

% Impresión de resultados
output ["Número de conejos: ", show(x1), "\n"];
output ["Número de pollos: ", show(x2), "\n"];
output ["Beneficio máximo: ", show(beneficio), " pesetas\n"];
```

2.

```
% variables
var int: x1; % surtidos del tipo 1
var int: x2; %  surtidos del tipo 2

% Función objetivo (beneficio)
var int: beneficio = 450*x1 + 560*x2;

% Restricciones
constraint 150*x1 + 200*x2 <= 200000; % Disponibilidad de polvorones
constraint 80*x1 + 100*x2 <= 104000; % Disponibilidad de mantecados
constraint x1 + x2 <= 1200; % Disponibilidad de cajas
% Restricciones de no negatividad
constraint x1 >= 0;
constraint x2 >= 0;

%  maximizar
solve maximize beneficio;

% Impresión de resultados
output ["Número de surtidos del tipo 1: ", show(x1), "\n"];
output ["Número de surtidos del tipo 2: ", show(x2), "\n"];
output ["Beneficio máximo: ", show(beneficio), " pesetas\n"];
```

3.

```
%  variables
var int: x1; %  sillas
var int: x2; %  mesas

% Función objetivo (beneficio)
var int: beneficio = x1 + 2*x2;

% Restricciones
constraint x1 + 3*x2 <= 9; % Disponibilidad de horas en la sección de montaje
constraint 2*x1 + x2 <= 8; % Disponibilidad de horas en la sección de pintura
```

```
% Restricciones de no negatividad
constraint x1 >= 0;
constraint x2 >= 0;

% maximizar
solve maximize beneficio;

% Impresión de resultados
output ["Número de sillas: ", show(x1), "\n"];
output ["Número de mesas: ", show(x2), "\n"];
output ["Beneficio máximo: ", show(beneficio), "\n"];
```

4.

```
%  variables
var int: x1; % unidades  tipo A
var int: x2; % unidades  tipo B
var int: x3; %  unidades  tipo C

% Función objetivo (beneficio)
var int: beneficio = 4000*x1 + 3000*x2 + 2000*x3;

% Restricciones
constraint 3*x1 + 3*x2 + x3 <= 24; % Disponibilidad de tiempo de mano de obra
constraint 6*x1 + 4*x2 + 3*x3 <= 60; % Disponibilidad de tiempo de revisión
constraint x1 + x2 + x3 <= 12; % Restricción de número de herramientas
% Restricciones de no negatividad
constraint x1 >= 0;
constraint x2 >= 0;

%  maximizar
solve maximize beneficio;

% Impresión de resultados
output ["Número de herramientas A: ", show(x1), "\n"];
output ["Número de herramientas B: ", show(x2), "\n"];
output ["Número de herramientas C: ", show(x3), "\n"];
output ["Beneficio máximo: ", show(beneficio), " pesetas\n"];
```

5.

```
% variables
var int: x1; % endodoncias
var int: x2; % sesiones de estomatología general

% Función objetivo (beneficio)
var int: beneficio = 5000*x1 + 4000*x2;

% Restricciones
constraint 0.75*x1 + 0.75*x2 <= 16; % Disponibilidad de tiempo de sillón
constraint 1.5*x1 + x2 <= 24; % Disponibilidad de tiempo de asistentes
constraint 0.25*x1 + 0.5*x2 <= 8; % Disponibilidad de tiempo del dentista
% Restricciones de no negatividad
constraint x1 >= 0;
constraint x2 >= 0;

%  maximizar
solve maximize beneficio;

% Impresión de resultados
```

```
output ["Número de endodoncias: ", show(x1), "\n"];
output ["Número de sesiones de estomatología general: ", show(x2), "\n"];
output ["Beneficio máximo: ", show(beneficio), " pesetas\n"];
```

6.

```
% parámetros
param int: oferta_region_I = 120;
param int: oferta_region_II = 250;
param int: demanda_molino_A = 200;
param int: demanda_molino_B = 150;

% Costos de transporte por tonelada
param int: costo_I_A = 5;
param int: costo_I_B = 4;
param int: costo_II_A = 5;
param int: costo_II_B = 6;

% Definición de variables de decisión
var int: x1_A; % Toneladas transportadas de la región I al molino A
var int: x1_B; % Toneladas transportadas de la región I al molino B
var int: x2_A; % Toneladas transportadas de la región II al molino A
var int: x2_B; % Toneladas transportadas de la región II al molino B

% Función objetivo: minimizar el costo total de transporte
var int: costo_total = costo_I_A * x1_A + costo_I_B * x1_B +
                       costo_II_A * x2_A + costo_II_B * x2_B;

% Restricciones
constraint x1_A + x1_B <= oferta_region_I; % Oferta de la región I
constraint x2_A + x2_B <= oferta_region_II; % Oferta de la región II
constraint x1_A + x2_A >= demanda_molino_A; % Demanda del molino A
constraint x1_B + x2_B >= demanda_molino_B; % Demanda del molino B

% Restricciones de no negatividad
constraint x1_A >= 0;
constraint x1_B >= 0;
constraint x2_A >= 0;
constraint x2_B >= 0;

% Minimizar la función objetivo
solve minimize costo_total;

% Imprimir resultados
output ["Costo mínimo de transporte: ", show(costo_total), "\n"];
output ["Toneladas transportadas de la región I al molino A: ", show(x1_A), "\n"];
output ["Toneladas transportadas de la región I al molino B: ", show(x1_B), "\n"];
output ["Toneladas transportadas de la región II al molino A: ", show(x2_A), "\n"];
output ["Toneladas transportadas de la región II al molino B: ", show(x2_B), "\n"];
```

7.

```
%  parámetros
param int: calorias_A = 1000;
param int: calorias_B = 2000;
param int: proteinas_A = 25;
param int: proteinas_B = 100;
param int: precio_A = 60;
param int: precio_B = 210;
```

```
param int: calorias_min = 3000;
param int: proteinas_min = 100;

% Definición de variables de decisión
var int: x_A; % Kilogramos de alimento A
var int: x_B; % Kilogramos de alimento B

% Función objetivo: minimizar el costo total de la dieta
var int: costo_total = precio_A * x_A + precio_B * x_B;

% Restricciones
constraint calorias_A * x_A + calorias_B * x_B >= calorias_min;
constraint proteinas_A * x_A + proteinas_B * x_B >= proteinas_min;

% Restricciones de no negatividad
constraint x_A >= 0;
constraint x_B >= 0;

% Minimizar la función objetivo
solve minimize costo_total;

% Imprimir resultados
output ["Costo mínimo de la dieta: ", show(costo_total), " pesetas\n"];
output ["Kilogramos de alimento A: ", show(x_A), " kg\n"];
output ["Kilogramos de alimento B: ", show(x_B), " kg\n"];
```

8.

```
% Definición de parámetros
param int: calorias_A = 1000;
param int: calorias_B = 2000;
param int: proteinas_A = 25;
param int: proteinas_B = 100;
param int: precio_A = 60;
param int: precio_B = 210;
param int: calorias_min = 3000;
param int: proteinas_min = 100;

% Definición de variables de decisión
var int: x_A; % Kilogramos de alimento A
var int: x_B; % Kilogramos de alimento B

% Función objetivo: minimizar el costo total de la dieta
var int: costo_total = precio_A * x_A + precio_B * x_B;

% Restricciones
constraint calorias_A * x_A + calorias_B * x_B >= calorias_min;
constraint proteinas_A * x_A + proteinas_B * x_B >= proteinas_min;

% Restricciones de no negatividad
constraint x_A >= 0;
constraint x_B >= 0;

% Minimizar la función objetivo
solve minimize costo_total;

% Imprimir resultados
output ["Costo mínimo de la dieta: ", show(costo_total), " pesetas\n"];
output ["Kilogramos de alimento A: ", show(x_A), " kg\n"];
```

```
output ["Kilogramos de alimento B: ", show(x_B), " kg\n"];
```