Programowanie liniowe.

Zadanie 1.

W Bistro Oliwka można kupić 3 różne produkty: kanapki, słodycze, pierogi. Każdy student stara się zbilansować swoją dietę (zjeść odpowiednią liczbę kanapek, słodyczy i pierogów), tak aby dostarczyć mózgowi odpowiednią ilość składników odżywczych, jak białko, tłuszcze, witaminy i węglowodany. Normy mówią, że każdy student powinien otrzymać co najmniej 250 jednostek białka, 60 jednostek tłuszczu, 100 jednostek witamin oraz 220 jednostek węglowodanów. Normy mówią również, że nie powinno zjadać się więcej niż zalecana dawka. Znamy zawartość poszczególnych składników w produktach oraz ceny jednostkowe produktów. Informacje te są zebrane w tabeli. Należy ustalić ile i których produktów pownienien zjeść student, aby jego organizam działał jak należy, a koszt diety był minimalny.

Składniki	Produkty			Minimalna ilość składnika
	kanapki	pierogi	słodycze	
białko	4	6	15	250
tłuszcz	2	2	0	60
witaminy	5	3	4	100
węglowodany	7	3	12	220
Cena (PLN)	2	1,5	3	

Zadanie 2.

Do produkcji stołów i krzeseł firma ST&KA.SA zużywa drewno, skórę oraz klej. Wyprodukowanie każdego z produktów związane jest z odpowiednim nakładem pracy. Wszystkie informacje zawarto w tabeli. Jako nowy manager (którego prowizja jest związana z zyskiem firmy) zaplanuj ile stołów i krzeseł należy wyprodukować, żeby osiągnąć maksymalny zysk.

Zasoby	Produkcja		Ograniczenia na zasoby
	krzesło	stół	
drewno	5	25	500
skóra	0,5	-	15
klej	100	250	7500
Nakład pracy	10	10	400
Zysk	100	200	

UWAGA:
Aby znaleźć rozwiązanie proszę posłużyć się funkcją `linprog` z Matlaba. Odkrycie zawiłości jej składni i napisanie odpowiedniego skryptu jest podstawą oceny.

linprog

https://uk.mathworks.com/help/optim/ug/linprog.html

Solve linear programming problems

Syntax

```
x = linprog(f,A,b)
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,options)
x = linprog(problem)
[x,fval] = linprog(___)
[x,fval,exitflag,output] = linprog(___)
[x,fval,exitflag,output,lambda] = linprog(___)
```

Description

Linear programming solver

Finds the minimum of a problem specified by

$$\min_x f^T x \text{ such that } \begin{cases} A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub. \end{cases}$$

f, x, b, beq, lb, and ub are vectors, and A and Aeq are matrices.

> **Note**
>
> `linprog` applies only to the solver-based approach. For a discussion of the two optimization approaches, see First Choose Problem-Based or Solver-Based Approach.

`x = linprog(f,A,b)` solves min `f'*x` such that `A*x ≤ b`. example

`x = linprog(f,A,b,Aeq,beq)` includes equality constraints `Aeq*x = beq`. Set `A = []` and `b = []` if no inequalities exist. example

`x = linprog(f,A,b,Aeq,beq,lb,ub)` defines a set of lower and upper bounds on the design variables, `x`, so that the solution is always in the range `lb ≤ x ≤ ub`. Set `Aeq = []` and `beq = []` if no equalities exist. example

> **Note**
>
> If the specified input bounds for a problem are inconsistent, the output `fval` is `[]`.

`x = linprog(f,A,b,Aeq,beq,lb,ub,options)` minimizes with the optimization options specified by `options`. Use `optimoptions` to set these options. example

`x = linprog(problem)` finds the minimum for `problem`, a structure described in `problem`. example

You can import a `problem` structure from an MPS file using `mpsread`. You can also create a `problem` structure from an `OptimizationProblem` object by using `prob2struct`.

`[x,fval] = linprog(___)`, for any input arguments, returns the value of the objective function `fun` at the solution `x`: `fval = f'*x`. example

`[x,fval,exitflag,output] = linprog(___)` additionally returns a value `exitflag` that describes the exit condition, and a structure `output` that contains information about the optimization process. example

`[x,fval,exitflag,output,lambda] = linprog(___)` additionally returns a structure `lambda` whose fields contain the Lagrange multipliers at the solution `x`. example