Exercise 1

In a dark, dingy alleyway you meet a mysterious shadowy figure. It seems you have taken a wrong turn home, but you cannot remember where you came from. For some reason, you know the figure means no harm, but it feels unwise to anger it. You cannot fully comprehend its figure, nor can you count its number of limbs. One of its appendages points towards your pocket. Now, feeling a cold shiver down your spine, you feel in your pocket. You find a crumpled note in your hand, depicting some rune-like script you haven't seen before. The note is oddly captivating, and the runes seem to dance over the paper without moving. After seconds that feel like minutes and minutes that feel like seconds, the note begins to speak its contents to you, without making a single sound.

- Six items: brimstone, arsenicum album, wormwood, nightshade, cinnabar and phlogiston.
- Costs per gram: 30, 10, 5, 20, 74 and 101.
- Energy per gram: 90, 2, 4, 8, 50 and 320.
- Corruption per gram: 234, 15, 0, 23, -4 and 20.
- Sulphur per gram: 30, 32, 2, 8, 0 and 50.
- Salt per gram: 60, 45, 12, 9, 35 and 0.
- Mercury per gram: 0, 24, 5, 12, 34 and 0.
- Between 460 and 500 energy.
- At most 616 corruption.
- Salt should outweigh the sum of mercury and sulphur.
- Mercury should be at most two-thirds of energy.
- Sulphur should exceed corruption by at least 30.
- Goal: minimize costs

a) Formulate the spectre's diet problem as a linear program. Introduce the decision variables clearly, and state the objective, constraints and variable bounds. Explain each of them.

Decision Variables:

- $x_1, x_2, x_3, x_4, x_5, x_6$ as the grams of items in required in the diet: brimstone, arsenicum album, wormwood, nightshade, cinnabar, and phlogiston respectively

Objective:

Minimize the costs of the diet's recipe

Parameters:

Cost per gram: $30x_1 , 10x_2 , 5x_3 , 20x_4 , 74x_5 , 101x_6$

Energy per gram: $90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6$

Corruption per gram: $234x_1 + 15x_2 + 0x_3 + 23x_4 - 4x_5 + 20x_6$

Sulphur per gram: $30x_1 + 32x_2 + 2x_3 + 8x_4 + 50x_6 + 0x_5$

Salt per gram: $60x_1 + 45x_2 + 12x_3 + 9x_4 + 35x_5 + 0x_6$

Mercury per gram: $0x_1 + 24x_2 + 5x_3 + 12x_4 + 34x_5 + 0x_6$

Constraints:

1. Energy Requirement -> $460 < \quad < 500$

$90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6 > \quad 460$

$90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6 < 500$

2. Corruption Limit -> ≤ 616

$234x_1 + 15x_2 + 0x_3 + 23x_4 - 4x_5 + 20x_6 \leq 616$

3. Salt Dominance -> Salt > Mercury + Sulfure

$60x_1 + 45x_2 + 12x_3 + 9x_4 + 35x_5 + 0x_6 > 0x_1 + 24x_2 + 5x_3 + 12x_4 + 34x_5 + 0x_6 + 30x_1 + 32x_2 + 2x_3 + 8x_4 + 0x_5 + 50x_6$

$\Leftrightarrow 30x_1 + 19x_2 + 5x_3 - 11x_4 + 1x_5 - 50x_6 > 0$

4. Mercury Constraint -> Mercury $\leq$ ⅔ Energy

$0x_1 + 24x_2 + 5x_3 + 12x_4 + 34x_5 + 0x_6 \leq \frac{2}{3} * (90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6)$

5. Sulphur Constraint -> Sulfure $\geq$ Corruption + 30

$30x_1 + 32x_2 + 2x_3 + 8x_4 + 0x_5 + 50x_6 \geq 234x_1 + 15x_2 + 0x_3 + 23x_4 - 4x_5 + 20x_6 + 30$

$\Leftrightarrow -204x_1 + 17x_2 + 2x_3 - 15x_4 + 4x_5 + 30x_6 \geq 30$

Variable Bounds:

A quantity (gram) of each item included in the diet can be or Null or positive but not negative

$x_i \geq 0$ for $i \in \{1, 2, 3, 4, 5, 6\}$

Objective Function:

min $30x_1+10x_2+5x_3+20x_4+74x_5+101x_6$

$90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6 > 460$

$90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6 < 500$

$234x_1 + 15x_2 + 0x_3 + 23x_4 - 4x_5 + 20x_6 \leq 616$

$60x_1 + 45x_2 + 12x_3 + 9x_4 + 35x_5 + 0x_6 > 0x_1 + 24x_2 + 5x_3 + 12x_4 + 34x_5 + 30x_1 + 32x_2 + 2x_3 + 8x_4 + 0x_5 + 50x_6$

$0 + 24x_2 + 5x_3 + 12x_4 + 34x_5 \leq \frac{2}{3} \text{ x } (90x_1 + 2x_2 + 4x_3 + 8x_4 + 50x_5 + 320x_6)$

$30x_1 + 32x_2 + 2x_3 + 8x_4 + 0x_5 + 50x_6 \geq 234x_1 + 15x_2 + 0x_3 + 23x_4 - 4x_5 + 20x_6 + 30$

$x_i \geq 0$ for $i \in \{1, 2, 3, 4, 5, 6\}$

A few weeks later, you read in the newspaper that there has been a series of break-ins at the local pharmacy, hardware store and florist. The paper additionally lists the items stolen:

- 0.1 grams of brimstone;
- 5 grams of cinnabar;
- 2 flowers nightshade, 0.5 grams each;
- 1 bundle of wormwood at 10 grams;
- 0.5 grams of phlogiston;
- 4 grams of arsenicum, of which 25% is of the album variant;

You ponder whether the spectre actually used your model, so you walk over to you notepad and start making calculations.

b) Is the apparent diet of the spectre feasible? In other words, does it adhere to its own restrictions? Explain your answer by using calculations.

The amounts stolen:

- Brimstone: 0.1 grams ($x_1 = 0.1$)

- Arsenicum album: 4 grams, 25% is album variant ($x_2 = 1$)
- Wormwood: 1 bundle, 10 grams ($x_3 = 10$)
- Nightshade: 2 flowers, 0.5 grams each ($x_4 = 1$)
- Cinnabar: 5 grams ($x_5 = 5$)
- Phlogiston: 0.5 grams ($x_6 = 0.5$)

Check if the stolen amount respect the constraints:

1. Energy:

$90(0.1) + 2(1) + 4(10) + 8(1) + 50(5) + 320(0.5) = 9 + 2 + 40 + 8 + 250 + 160 = 469$

$460 < 469 < 500$ This is within the range 460 to 500, the constraint is respected.

2. Corruption:

$234(0.1) + 15(1) + 0(10) + 23(1) - 4(5) + 20(0.5) = 23.4 + 15 + 0 + 23 - 20 + 10 = 51.4$

$51.4 < 616$ This is less than 616, the constraint is respected.

3. Salt vs. Mercury and Sulphur:

Salt: $60(0.1) + 45(1) + 12(10) + 9(1) + 35(5) = 6 + 45 + 120 + 9 + 175 = 355$

Sulphur: $30(0.1) + 32(1) + 2(10) + 8(1) + 50(0.5) = 3 + 32 + 20 + 8 + 25 = 88$

Mercury: $0 + 24(1) + 5(10) + 12(1) + 34(5) + 0 = 0 + 24 + 50 + 12 + 170 = 256$

$355 > 88 + 256 = 344$ The constraint is respected.

4. Mercury:

$256 \leq \frac{2}{3} * 469 = 312.67$

$256 < 312.67$ The constraint is respected.

5. Sulphur:

$88 \geq 51.4 + 30 = 81.4$

$88 > 81.4$ The constraint is respected.

With the amount stolen from each item, the spectre's diet is feasible since it answers all constraints.

And the total costs of this Diet would be as follows:

$30*0.1+10*1+5*10+20*1+74*5+101*0.5 = 503.5$

c) If the diet in b) was feasible, can you find a cheaper feasible diet? Otherwise, if the diet in b) was not feasible, can you find a feasible diet instead?

To find a cheaper feasible diet we can solve the linear program we can generate an R function that gave the following results:

The R function to solve the fixed model optimised the transportation costs, with an objective value of 188.7779.

Decision Variables:

- $x_1 = 0.1505058$
- $x_2 = 0$
- $x_3 = 11.59882$
- $x_4 = 0$
- $x_5 = 0$
- $x_6 = 0.250185$

The optimal cost is thus:

$0.1505058 * 30 \simeq 4.515174$

$0 * 10 = 0$

$11.59882 * 5 \simeq 57.9941$

$0 * 20 = 0$

$0 * 74 = 0$

$1.250185 * 101 \simeq 126.268685$

Total cost = 188.7779

Exercise 2

As if someone had died, your boss slams the door to your office open. Before the deafening sound of the door even reaches your ears, he starts bawling in your arms. After six straight minutes of incomprehensible sobbing and some awkward headpats from your side, you can finally make some sense of his words.

Apparently, his second favourite optimization goon, John Johnson von Johnsonschneider the IV, hadn't paid enough attention during his studies. The problem lies in the "Super-Important Transportation Problem" (SITP) that generates around 94% ($\pm$10%) of the company's profits. Optimizing that problem each day was John's task, but he only implemented January 23rd, 2021 and then took a vacation until now. Just 4 minutes ago, the boss was informed that the company is losing significant money due to doing the same thing every day and ignoring all inputs.

The boss looks you straight in the eyes, his eyes still puffy and swollen from his emotional outburst. His snot trickles slowly down his upper lip, only changing direction slightly due to his moustache hairs diverting the stream like a pachinko board. "You must save my company," he exclaims, while unidentifiable liquids exit his mouth and enter your personal space, "please generalize this model!"

He storms out of the office. Still baffled, you receive an email mere seconds later, containing the details of the SITP. A single tear slowly emerges in your right eye and drips down on your hand. Overwork it is then. Work-life balance is overrated.

The email shows you the model that JJvJ IV had constructed:

$$
\begin{aligned}
\min\ & 14x_{1A} + 13x_{1B} + 10x_{1C} + 19x_{1D} \\
& +15x_{2A} + 19x_{2B} + 14x_{2C} + 19x_{2D} \\
& +20x_{3A} + 18x_{3B} + 13x_{3C} + 14x_{3D} \\
\text{s.t.}\quad & x_{1A} + x_{2A} + x_{3A} \geq 66 \\
& x_{1B} + x_{2B} + x_{3B} \geq 62 \\
& x_{1C} + x_{2C} + x_{3C} \geq 88 \\
& x_{1D} + x_{2D} + x_{3D} \geq 74 \\
& x_{1A} + x_{1B} + x_{1C} + x_{1D} \leq 150 \\
& x_{2A} + x_{2B} + x_{2C} + x_{2D} \leq 125 \\
& x_{3A} + x_{3B} + x_{3C} + x_{3D} \leq 100 \\
& x_{1A}, x_{1B}, x_{1C}, x_{1D}, x_{2A}, x_{2B}, x_{2C}, x_{2D}, x_{3A}, x_{3B}, x_{3C}, x_{3D} \geq 0
\end{aligned}
$$

The email states that there are 3 production facilities, "1", "2" and "3", and 4 shipment locations, "A", "B", "C" and "D". Apparently, the company must decide how much to produce in each facility and to which location it should be shipped to. For instance, the variable x_{2B} states how much is shipped from production facility 2 to location B.

a) Explain the objective and different constraints. Try to give them an interpretable meaning based on the problem context.

Objective:

Minimizing the transportation cost between facilities and locations. So minimize the total transportation costs.

$\min\ 14x_{1A} + 13x_{1B} + 10x_{1C} + 19x_{1D} + 15x_{2A} + 19x_{2B} + 14x_{2C} + 19x_{2D} + 20x_{3A} + 18x_{3B} + 13x_{3C} + 14x_{3D}$

Each variables represent the cost of shipping (the coefficients) products from one facility (1, 2 or 3) to a shipment location (A, B, C or D).

Constraints:

- The first constraints can be interpreted as demand constraints. It is needed to ensure that each location has enough products to answer its demand.

$x_{1A} + x_{2A} + x_{3A} \geq 66$

$x_{1B} + x_{2B} + x_{3B} \geq 62$

$x_{1C} + x_{2C} + x_{3C} \geq 88$

$x_{1D} + x_{2D} + x_{3D} \geq 74$

- The next constraints can be interpreted as production capacity constraints. It is needed to ensure that each facility produces up to a maximum capacity of products.

$x_{1A} + x_{1B} + x_{1C} + x_{1D} \leq 150$

$x_{2A} + x_{2B} + x_{2C} + x_{2D} \leq 125$

$x_{3A} + x_{3B} + x_{3C} + x_{3D} \leq 100$

- the last constraint is the non-negativity constraint. A quantity cannot be negative, we need to ensure that the number of products shipped and produced are Null or positive.

$x_{ij} \geq 0$ for all $i \in \{1, 2, 3\}, j \in \{A, B, C, D\}$

b) Remove the numbers from the formulation by introducing appropriate parameters for them. State the interpretation of each parameter. Note: you can leave the number of facilities to 3 and the number of locations to 4.

Introducing parameters to replace the number in the formulation:

first let's replace the letters of each location to numbers

- C_{ij} : Cost per unit shipped from facility i to location j, $i \in \{1, 2, 3\}, j \in \{1, 2, 3, 4\}$
- d_j : Demand at location j, $j \in \{1, 2, 3, 4\}$
- s_i : Supply capacity of facility i, $i \in \{1, 2, 3\}$

Reformulating the constraints with the parameters:

- Demand constraint:

$$\sum_{i=1}^{3} x_{ij} \geq d_j$$

- Supply constraint:

$$\sum_{j=1}^{4} x_{j} \geq s_i$$

- The non-negativity constraint stays the same

Reformulating the objective function:

$$\sum_{i=1}^{3}\sum_{j=1}^{4} x_{ij}\, c_{ij}$$

c) Write an R function to solve the fixed model (so with the numbers). Report its objective value and which values the decision variables take.

The R function to solve the fixed model optimised the transportation costs, with an objective value of 3712 :

- $x_{1A}=0$
- $x_{1B}=62$
- $x_{1C}=88$
- $x_{1D}=0$
- $x_{2A}=66$
- $x_{2B}=0$
- $x_{2C}=0$
- $x_{2D}=0$
- $x_{3A}=0$
- $x_{3B}=0$
- $x_{3C}=0$
- $x_{3D}=74$

The optimal shipment quantities are thus:
From Facility 1 to Location B: 62 units
From Facility 1 to Location C: 88 units
From Facility 2 to Location A: 66 units
From Facility 3 to Location D: 74 units

d) Describe one way to solve this problem greedily. Execute this greedy algorithm (either by hand or by computer) and compare the solution to the one you obtained by c)

To solve the problem greedily, one can create a cost matrix, selecting the minimum cost cell and answering the demand of the location this cell belongs to by taking into account the supply limitations.
After the minimum cost cell has been dealt with, move on to the next minimum cost cell and reiterate the process explained above until all demand of each location is satisfied and at a minimum cost while answering the supply constraints of each facility.

- Costs matrix :

	A	*B*	*C*	*D*
1	**14**	**13**	**10**	**19**
2	**15**	**19**	**14**	**19**
3	**20**	**18**	**13**	**14**

- Demand: A: 66, B: 62, C: 88, D: 74
- Supplie: 1: 150, 2: 125, 3: 100

Minimum cost cell
The minimum cost is 10 (1;C)
Production capacity in facility 1: 150
-> Ship 88 units from facility 1 to location C
Remaining demand for C: 88 - 88 = 0
Remaining supply for facility 1: 150 - 88 = 62

Next minimum cost cell
next minimum is 13 (1;B)
Production capacity left in facility 1: 62
-> Ship 62 units from facility 1 to location B
Remaining demand for B: 62 - 62 = 0
Remaining supply for facility 1: 62 - 62 = 0

Next minimum cost cell
next minimum is 14 (3;D)
Production capacity left in facility 3: 100
-> Ship 74 units from facility 3 to location D.
Remaining demand for D: 74 - 74 = 0
Remaining supply for facility 3: 100 - 74 = 26

Next minimum cost cell
next minimum cost is 15 (A;2)
Production capacity left in facility 2: 125
-> Ship 66 units from facility 2 to location A.
Remaining demand for A: 66 - 66 = 0
Remaining supply for facility 2: 125 - 66 = 59

Final Allocation Matrix:

	A	*B*	*C*	*D*
1	**0**	**62**	**88**	**0**
2	**66**	**0**	**0**	**0**
3	**0**	**0**	**0**	**74**
D_J	66	62	88	74

Final Cost Calculation:

1 to C: 88*10 = 880
1 to B: 62*13 = 806
3 to D: 74*14 = 1036
2 to A: 66*15 = 990
Total cost: 880 + 806 + 990 + 1036 = 3712

This is the same solution as the function in R provided which means that the greedy approach is the most optimal solution and does not need to be improved

Exercise 3

Binary knapsack, with two constraints. “Wait a minute,” you exclaim, “haven’t I seen this one already, but it had only one constraint?” Well, that could only be true if you have attended the second lecture. After all, this problem is not present in the first lecture. “But isn’t that kind of lame?” Perhaps, but I created this assignment, so I decide which problems go into it. Besides, you might like its familiarity, you know? You scoff at this fourth-wall-breaking atrocity. First, this pompous writer wastes your time by creating elaborate nonsense for the first two exercises, and now it tries to create some kind of half-baked “story” for the third exercise. Unbelievable, inconceivable, preposterous. Unfortunately, the writer could not care less, and so did just scam you in reading this. You’ll never get those seconds back.
In any case, the generic model is given below:

$$\max \sum_{i \in I} p_i x_i$$

$$\text{s.t.} \quad \sum_{i \in I} w_i x_i \leq W$$

$$\sum_{i \in I} v_i x_i \leq V$$

$$x_i \in \mathbb{B} \qquad \forall i \in I$$

Here, I is the set of items we can choose, pi $>$ 0 is the profit of item i, wi is the weight of item i, W is the weight limit, vi is the volume of item i and V is the volume limit of item i. The decision variable xi tells whether we put item i in the knapsack (xi $=$ 1) or not (xi $=$ 0). Hint: xi $\in$ B is one way to state that the variables xi are binary, i.e., either 0 or 1.
p_i : profit of i
w_i : weight of i
v_i : volume of i
W: weight limit of the knapsack
V: volume limit of the knapsack
x_i : if item i is in the knapsack ($x_i = 1$ or 0 if not)

a) Implement the above model in R and solve for all 30 instances in the file knapsack.txt found on Canvas. Each line of the file is a single instance of the form

$$W, V, n, (p_1, w_1, v_1), (p_2, w_2, v_2), (p_3, w_3, v_3), \ldots, (p_n, w_n, v_n)$$

where n is the number of items. In other words, the first 3 numbers on the line signify the weight limit W, the volume limit V and the number of items n respectively. Then, each triplet of numbers, enclosed in parenthesis, signifies one item’s profit, weight and volume respectively. For each instance, note the objective value, the weight of the chosen items versus the weight limit, the volume of the chosen items versus the volume limit and the chosen items.

The R function to solve this model for each instances (knapsacks) of the file, provided the following optimal results :

Instance index	optimal (max) profit	items chosen	total weight/ max weight	total volume/ max volume
1	754	1 2 3 6 7	56/81	13/13

2	576	1 2 4 5	59/60	11/13
3	759	2 3 4 6	75/77	15/17
4	684	4 5 10 11 13	62/63	12/12
5	820	2 3 5 6 9	50/54	18/19
6	941	2 3 5 8 9 11	71/90	14/14
7	888	3 5 6 9 10	58/61	12/12
8	640	1 3 4 7	75/82	13/13
9	1285	2 3 7 11 12 13 14 15	97/99	15/15
10	605	1 4 5 6	49/52	12/12
11	1116	1 2 3 4 6 10 11	84/86	16/18
12	695	4 6 7 8	61/62	12/13
13	1244	1 2 4 6 8 9 10	95/96	16/16
14	787	1 2 3 7 8	76/77	10/11
15	617	1 9 10 12	46/52	11/11
16	849	2 3 4 7 9	84/99	12/12
17	752	1 2 4 5	65/66	14/19
18	863	1 2 3 4 5 7	93/97	15/16
19	661	1 2 6 9	53/55	8/12
20	596	1 2 5 7	68/96	11/11
21	667	1 2 4 5	55/90	10/10
22	453	1 2 3 5	48/53	13/19
23	669	1 3 5 6	59/60	16/17
24	897	1 3 4 5 7	55/57	18/18
25	749	1 2 5 6 9	58/59	10/15
26	472	1 2 5	49/57	4/10
27	1029	2 4 5 6 13 15	82/83	16/16
28	994	1 2 3 8 13 14	72/76	16/16
29	980	4 7 10 11 12 15	68/69	19/19

30	796	1 2 3 4 5 6	95/95	18/19

b) Describe a greedy algorithm to solve this problem. Implement it in R and solve the same instances as in a). Compare the objective values.

A greedy algorithm for this problem with two constraints will be the following way:
- Initialize the solution
- Add 1 item that hasn't been added yet
- Choose the "best item"
  - ➢ The item with the highest profit-to-weight ratio p_i
  - ➢ The item with the highest profit-to-volume ratio p_i

And iteratively select items from the sorted list until either the weight limit or the volume limit is reached.
Then compare the profit of the two methods and chose the method with the highest profit as final optimal solution for each instance.

R output for the greedy algorithm compared to the one from the exact solution:

Instance index	optimal (max) profit	optimal profit, greedy	items chosen, greedy	total weight/ max weight, greedy	total volume/ max volume, greedy
1	754	754	1 2 3 6 7	56/81	13/13
2	576	576	1 2 4 5	59/60	11/13
3	759	759	2 3 4 6	75/77	15/17
4	684	670	4 7 10 11 12	57/63	12/12
5	820	820	2 3 5 6 9	50/54	18/19
6	941	941	2 3 5 8 9 11	71/90	14/14
7	888	837	3 5 6 7 10	61/61	10/12
8	640	640	1 3 4 7	75/82	13/13
9	1285	1285	2 3 7 11 12 13 14 15	97/99	15/15
10	605	573	2 4 5 6	42/52	10/12
11	1116	1070	1 2 3 4 9 10 11	75/86	17/18
12	695	695	4 6 7 8	61/62	12/13
13	1244	1244	1 2 4 6 8 9 10	95/96	16/16
14	787	787	1 2 3 7 8	76/77	10/11
15	617	532	1 10 13	47/52	8/11

16	849	849	2 3 4 7 9	84/99	12/12
17	752	752	1 2 4 5	65/66	14/19
18	863	863	1 2 3 4 5 7	93/97	15/16
19	661	661	1 2 6 9	53/55	8/12
20	596	578	1 3 5 7	59/96	10/11
21	667	667	1 2 4 5	55/90	10/10
22	453	453	1 2 3 5	48/53	13/19
23	669	668	1 2 6 8	57/60	15/17
24	897	897	1 3 4 5 7	55/57	18/18
25	749	713	4 5 6 10	56/59	10/15
26	472	472	1 2 5	49/57	4/10
27	1029	1029	2 4 5 6 13 15	82/83	16/16
28	994	915	1 2 3 5 8 12	64/76	16/16
29	980	966	1 7 10 11 12 15	66/69	19/19
30	796	796	1 2 3 4 5 6	95/95	18/19

The green cells are the instances for which the greedy algorithm predicted the actual most optimal solution as was computed exactly for the previous question. In more than half of the cases, the optimal solution with the greedy method is the most optimal solution with the highest profit in general.

c) You can increase the bag's weight limit by K for a cost (negative profit) of C. Adjust the model to incorporate this constraint. If you introduce new decision variables or parameters, please introduce them clearly. You do not have to solve any instances.

We could introduce a new binary variable "y" that accounts for this option to increase the weight limit to a cerain cost. y -> binary variable (=1 if we increase the weight limit, =0 if not)

The new objective function would look like the following:

$$max \sum_i pi * xi - y * C$$

$$\sum_i (wi * xi) - y * K < W$$

$$\sum_i vi * xi < V$$

$$xi \in \{0,1\} \qquad y \in \{0,1\} \qquad \forall i \in I$$

Exercise 4

It seems as if you are finally done with your work for the week. No more spectres on the way home, no more sobbing bosses and certainly no more fourth wall breaks. Instead, you travel home (in a totally, 100% normal very human way), kick off your shoes (who wears shoes in their home?) and fall in your couch. You spend a few minutes daydreaming what to do with your well-deserved break. You check your phone, and see that, for some reason or another, a given number of your friends have contacted you with their plans for the coming days. Each friend suggests a number of activities for you to do together. Luckily, they are all flexible on the dates, so you can determine that yourself. You quickly notice that most of these suggestions contain significant overlap with other suggestions, and you, a person with high standards, don't get any additional enjoyment out of doing an activity multiple times. Furthermore, the number of friends that have contacted you greatly outnumbers the number of free days you have.
So you ponder. You want to maximize the number of unique activities done, while having a limited number of days to plan your friends in. "Hmmm", you think, "I could probably generalize this to any number of days, an arbitrary number of friends and arbitrary activity lists of each friend. Perhaps if I write down..." You interrupt yourself with a horrifying revelation.
Oh no.
Your phone takes off its disguise to reveal yet another exercise! The horror! Seems like your weekend is still far away, how cruel. Anyway, in this problem, you are given a number of sets. Each set contains some elements. Your task is to select k sets such that the number of unique elements in the sets is maximized. For instance, if the sets are

$$S_1 = \{0, 1, 2\}, S_2 = \{1, 2, 3\}, S_3 = \{2, 3, 4\}$$

then with k = 2 you would select S_1 and S_3, as then the number of unique elements in these sets is 5, namely 0, 1, 2, 3 and 4. S_n corresponds to each ≠ set and each sets corresponds to a list of activities between 0 to 4 all numbered from 1 to 20.

a) Describe how a greedy algorithm for this problem would work.

This is a greedy algorithm for this problem:
We start with a first set with the most elements, add other sets such as the number of chosen sets is less than k:

- and for each remaining set, calculate the number of new unique elements it would add if selected
- Select the set that adds the most new unique elements to the chosen sets

Add the selected set to the chosen sets and update the set of unique elements.
Continue until k sets are selected. You will maximize the total number of unique elements

b) Execute your greedy algorithm by implementing it in R on the instances in set Problem.txt on Canvas. Each line of the file is a single instance of the form n, k, S_1, S_2, . . . , S_n where n is the total number of sets to choose from, k is the number of sets to pick and Si is the contents of set i , denoted within curly braces as { e_{i1}, . . . , $e_{in\,i}$}. For each instance, note the number of unique items in the chosen sets, as well as the chosen sets themselves.

R output for the greedy algorithm:

Instance	Chosen Sets	Number of Unique Elements
1	{2, 8, 10, 15, 16} {4, 6, 9, 13, 14} {3, 20} {4, 5, 11, 13, 16} {13, 16, 17} {3, 4, 8, 9, 18}	16
2	{1, 7, 8, 10, 16} {2, 6, 12, 17} {2, 3, 9, 13}	12
3	{12, 14, 15, 17, 20} {11, 13, 16, 18} {3, 5, 6, 16} {4, 8, 15} {9, 11, 16} {15, 16, 18, 19}	16
4	{4, 8, 10, 11, 13} {2, 3, 12, 17, 18} {5, 7, 13, 15} {1, 3, 4, 20} {8, 9, 11, 19}	17
5	{2, 9, 12, 14, 20} {1, 6, 13, 15} {3, 9, 17, 18} {1, 10, 14, 15, 19}	14
6	{2, 4, 5, 13, 16} {1, 3, 16, 17, 20} {4, 9, 10, 12, 15}	13
7	{7, 9, 12, 13, 17} {11, 14, 15, 17, 18} {10, 16, 17, 20} {2, 5, 8, 9} {3, 7, 11, 12, 19}	17
8	{1, 3, 9, 10, 11} {1, 7, 8, 12, 18} {2, 8, 13, 15}	12
9	{2, 3, 10, 12, 19} {6, 8, 14, 15} {3, 6, 11, 13, 19} {4, 5, 10} {2, 9}	14
10	{7, 12, 15, 17, 18} {8, 9, 14, 20} {4, 11, 12, 13} {6, 10, 13} {1, 5, 9, 11}	16
11	{1, 8, 14, 18, 19} {2, 10, 14, 17, 20} {3, 5, 13, 15} {2, 3, 11, 12} {5, 6, 16, 17, 18} {4, 7, 10, 13, 14}	19
12	{2, 14, 16, 17, 18} {2, 5, 7, 8, 13} {16, 19, 20} {4, 5, 9} {2, 6, 7, 17} {10, 14, 17, 19}	15
13	{6, 7, 12, 16, 20} {1, 4, 9, 13, 20} {2, 5, 7, 10, 16} {1, 3, 6, 8, 17}	15
14	{1, 6, 9, 14, 17} {7, 12, 20} {1, 2, 3, 6, 11} {5, 9, 15, 17, 18} {4, 9, 14, 17, 19}	16
15	{5, 9, 13, 19, 20} {1, 2, 8, 11} {6, 13, 15, 17, 20} {10, 11, 16}	14
16	{2, 6, 7, 18, 20} {7, 8, 11, 15} {9, 12, 16} {1, 4, 15, 18} {2, 10, 11, 13}	15
17	{5, 9, 12, 18, 19} {8, 15, 16} {2, 10, 11} {3, 15, 19, 20}	13

18	{2, 7, 10, 18, 19} {8, 12, 20} {3, 10, 16, 18} {9, 10, 14}	12
19	{1, 4, 16, 17, 18} {3, 5, 7, 11, 12} {2, 6, 10, 17}	13
20	{1, 5, 10, 12, 20} {7, 14, 15, 17, 20} {1, 6, 8, 9} {2, 18, 19} {4, 5, 11, 14}	17
21	{2, 5, 9, 14, 15} {4, 10, 13, 17, 20} {6, 12, 14, 19, 20} {7, 8, 9, 13, 15}	15
22	{3, 14, 16, 19, 20} {1, 9, 12, 17} {5, 6, 7, 16, 18} {2, 9, 10, 13, 19} {1, 4, 5, 18} {5, 11, 12}	18
23	{2, 5, 11, 17, 18} {4, 6, 9, 20} {7, 8, 15, 18, 19} {1, 3, 5, 16, 18}	16
24	{4, 6, 7, 9, 11} {2, 3, 12, 15} {4, 8, 10, 14, 16} {4, 12, 13, 17, 20} {8, 19} {7, 14, 15, 16, 18}	18
25	{2, 4, 9, 11, 15} {1, 2, 5, 13, 14} {2, 10, 17, 19, 20} {8, 12, 19} {5, 16, 18, 20} {4, 5, 6, 7, 17}	19

c) Describe how a local search algorithm would work for this problem.

A local search algorithm for this problem would

Start with an initial solution, namely the one from the greedy algorithm.

- for each set in the current solution, consider replacing it with another set not currently in the solution
- Evaluate, calculate the number of unique elements for each potential new solution

Select if a neighboring solution improves the number of unique elements, move to this new solution. Repeat the neighborhood exploration until no further improvements can be found.

d) Implement your local search algorithm in R and execute it on the found solutions in b). Does your local search algorithm improve the found solutions? Report, for each instance, the number of unique items in the sets, as well as the chosen sets themselves. Also report whether this actually improves upon the solutions found in b).

R output for the local search algorithm compared to the one from the greedy algorithm:

Instance	Initial Number of Unique Elements	New Number of Unique Elements	Chosen Sets	Improvement
1	16	17	{2, 8, 10, 15, 16} {4, 6, 9, 13, 14} {6, 10, 16, 19, 20} {4, 5, 11, 13, 16} {13, 16, 17} {3, 4, 8, 9, 18}	Yes
2	12	13	{1, 7, 8, 10, 16} {5, 12, 18, 20} {2, 3, 9, 13}	Yes
3	16	17	{12, 14, 15, 17, 20} {1, 6, 13} {3, 5, 6, 16} {4, 8, 15} {9, 11, 16} {15, 16, 18, 19}	Yes

4	17	18	{8, 10, 14, 15} {2, 3, 12, 17, 18} {5, 7, 13, 15} {1, 3, 4, 20} {8, 9, 11, 19}	Yes
5	14	15	{2, 9, 12, 14, 20} {1, 2, 6, 11, 13} {3, 9, 17, 18} {1, 10, 14, 15, 19}	Yes
6	13	14	{2, 6, 14, 17, 18} {1, 3, 16, 17, 20} {4, 9, 10, 12, 15}	Yes
7	17	18	{6, 8, 9, 13, 15} {11, 14, 15, 17, 18} {10, 16, 17, 20} {2, 5, 8, 9} {3, 7, 11, 12, 19}	Yes
8	12	13	{1, 3, 9, 10, 11} {4, 7, 9, 14, 19} {2, 8, 13, 15}	Yes
9	14	15	{1, 10, 12} {6, 8, 14, 15} {3, 6, 11, 13, 19} {4, 5, 10} {2, 9}	Yes
10	16	17	{7, 12, 15, 17, 18} {8, 9, 14, 20} {2, 4, 15, 18, 20} {6, 10, 13} {1, 5, 9, 11}	Yes
11	19	20	{1, 8, 14, 18, 19} {2, 10, 14, 17, 20} {9, 15} {2, 3, 11, 12} {5, 6, 16, 17, 18} {4, 7, 10, 13, 14}	Yes
12	15	16	{2, 14, 16, 17, 18} {8, 13, 15} {16, 19, 20} {4, 5, 9} {2, 6, 7, 17} {10, 14, 17, 19}	Yes
13	15	16	{6, 15, 18, 20} {1, 4, 9, 13, 20} {2, 5, 7, 10, 16} {1, 3, 6, 8, 17}	Yes
14	16	18	{1, 8, 12, 13} {7, 12, 20} {1, 2, 3, 6, 11} {5, 9, 15, 17, 18} {4, 9, 14, 17, 19}	Yes
15	14	16	{4, 5, 11, 18, 19} {1, 2, 8, 11} {6, 13, 15, 17, 20} {9, 10, 12, 13}	Yes
16	15	16	{6, 12, 17, 20} {7, 8, 11, 15} {9, 12, 16} {1, 4, 15, 18} {2, 10, 11, 13}	Yes
17	13	14	{5, 9, 12, 18, 19} {4, 5, 12, 13, 16} {2, 10, 11} {3, 15, 19, 20}	Yes
18	12	13	{2, 7, 15, 19} {8, 12, 20} {3, 10, 16, 18} {9, 10, 14}	Yes
19	13	14	{1, 4, 8, 14, 20} {3, 5, 7, 11, 12} {2, 6, 10, 17}	Yes
20	17	18	{5, 12, 13, 16} {7, 14, 15, 17, 20} {1, 6, 8, 9} {2, 18, 19} {4, 5, 11, 14}	Yes
21	15	16	{5, 11, 18} {4, 10, 13, 17, 20} {6, 12, 14, 19, 20} {7, 8, 9, 13, 15}	Yes
22	18	19	3, 14, 16, 19, 20} {8, 12, 16, 17} {5, 6, 7, 16, 18} {2, 9, 10, 13, 19} {1, 4, 5, 18} {5, 11, 12}	Yes

23	16	17	{2, 8, 10, 11, 17} {4, 6, 9, 20} {7, 8, 15, 18, 19} {1, 3, 5, 16, 18}	Yes
24	18	19	{4, 6, 7, 9, 11} {2, 3, 12, 15} {5, 10, 15} {4, 12, 13, 17, 20} {8, 19} {7, 14, 15, 16, 18}	Yes
25	19	20	{2, 4, 9, 11, 15} {1, 2, 5, 13, 14} {3, 10, 11} {8, 12, 19} {5, 16, 18, 20} {4, 5, 6, 7, 17}	Yes

There is an improvement in every single instance from the greedy algorithm to the local search algorithm, this proves that the greedy algorithm provides optimal solutions but not always the most optimal.