CPS721: Assignment 3

Due: October 21, 2025, 9pm
Total Marks: 130 (worth 4% of course mark)
In groups of no more than 3

Late Policy: Late submissions are penalized by 3^n% when they are n days late.

Clarifications and Questions: Please use the discussion forum on the D2L site to ask questions. These will be monitored regularly. A Frequently Asked Questions (FAQ) page will also be created. Only ask questions by email if it is specific to your submission. Any questions asked with less than 12 hours left to the deadline are not likely to be answered.

PROLOG Instructions. Your code must run on ECLiPSe 7.1. If you cannot install this on your own machine, test it on the lab machines or the department moon servers.

You are NOT allowed to use ";" (disjunction), "!" (cut), "->" (if-then), `setof`, `bagof`, `findall`, or any other language features and library predicates that are not covered in class. Similarly, you should only use "=" and "is" for equality, and "not" for negation. Do NOT use "=:=", "\=", "+\", or other such commands. You will be penalized if you use these commands/features. You are only allowed to use ";" to get additional responses when interacting with PROLOG from the command line (this is equivalent to using the "More" button in the ECLiPSe GUI).

You ARE allowed to use the built-in `member` or `append` methods if they are useful to you.

Academic Integrity and Collaboration Policy: Any submission you make must be the work of your group and your group alone. You are also not allowed to post course materials (including assignments, slides, etc.) anywhere on the web, and you are also prohibited from using generative AI systems or online "help" websites for completing assignments.

Submission Instructions: You should submit ONE zip file called `assignment3.zip` on D2L. It should contain the following files:

```
assignment3_submission_info.txt
assignment3_report.txt
q1_8cust_generate_and_test.pl     q1_17cust_interleaving.pl
q2_network.pl
q3_kenken.pl
```

All these files have been given to you and you should fill them out using the format described. Ensure your names appear in all files — especially `assignment3_submission_info.txt` — and your answers appear in the file sections as requested. Do NOT edit, delete, or add any lines that begin with "`%%%%% SECTION:`". These are used by the auto-grader to locate the relevant code. You will lose marks if you edit such lines.

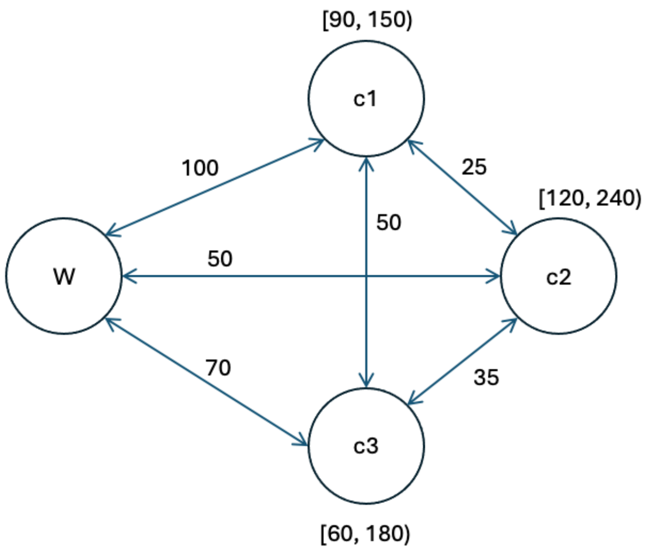
Your submission should not include any other files than those above. You will be penalized for using compression formats other than `.zip`. Submissions by email will not be accepted.

You may submit your solution multiple times. Doing so will simply overwrite your previous submission. We will therefore always mark the last submission your group makes.

1 The Delivery Truck Problem [50 marks]

Suppose you work for a delivery company that uses a single truck that starts at its warehouse, and then must visit each of a given set of customers in any order. Additionally, each customer has a specified window during which they are available. That is, the delivery truck must arrive at each customer within their specific time window in order to successfully make the delivery.

For example, consider the following image, which is a 3 customer problem:



Each circle represents a location, with `c1`, `c2`, and `c3` being the customers, and the warehouse labelled as a "W". The number on the lines between each location shows the time (in minutes) needed to travel between them. The figure also shows the time windows for each customer, which is given in terms of minutes after 9am. For example, the time window for `c1` is $[90, 150)$, meaning that the truck must arrive at `c1` between 10:30am (inclusive) and before 11:30am in order to successfully make the delivery to this customer. We will use this convention in the rest of the question.

A valid solution for this problem will have the truck go from the warehouse to `c3` (arriving at time 70), then to `c1` (arriving at 120), and finally to `c2` (arriving at 145). As can be seen, each arrival is within the customers specified time window. For example, the truck arrives at `c1` at time 120, which is in the time window $[90, 150)$. Additionally, we are making the following simplifying assumptions:

- We refer to all customers using the short form of `c1`, `c2`, `c3`, ...
- Deliveries happen instantaneously. Thus, if the truck leaves the warehouse at time 0, goes to `c2`, and then `c3`, it will arrive at `c3` at time 120, having successfully delivered to both `c2` and `c3`[1].
- The truck cannot "wait" anywhere. That is, it must immediately travel from one customer to the next.
- The truck cannot visit the same customer once. This means that in any valid solution, the one time that the truck visits a customer, it must be within their given time window.
- The truck does not need to return to the warehouse at the end. We only care about finding a valid ordering of the customers.[2].
- The truck cannot return to the warehouse in the middle of the trip. That is, once it leaves the warehouse, the truck should never return to the warehouse before it has visited each customer.

[1]In practice, delivery time can be incorporated by simply increasing the travel time by the expected delivery time.

[2]This might matter in a real problem if the delivery company's objective is to minimize fuel usage or total travel time

Your task is to create a set of Prolog programs to solve several instances of this problem. In particular, you will first consider using generate and test on different sizes of this program. Next you will use interleaving. We have provided the travel times and windows between the customers in several extra files. First, `q1.py` is a Python file which provides the travel times between customers and the time windows in Python lists. Technically, these lists can be used directly in your Prolog program by just copying them over. However, you may find it more convenient to add code to `q1.py` that will turn this data into a more directly useful KB that can be copied over into your Prolog files[3]. If you do not like Python, we have alternatively given the distances and time windows in two CSV files (`travel_times.csv` and `time_windows.csv`), such that you can use any language of your choice for the purpose of generating a KB. See `README_CSV` for more information on these files. These CSV files just duplicate the date in `q1.py`, so you should not need both the CSV files and the Python file. Note, you do NOT need to submit any code for how you generated your KB, just the final Prolog solutions themselves.

You will also be documenting your output and runtimes for this question — as well as questions 2 and 3 — in a report named `assignment3_report.txt`.

HINT: This task involves a number of similar constraint checks. In such cases, it is often useful to define helper predicates so you can avoid repetitive code.

a. [25 marks] The first task is to write programs for solving the given instance of this problem using generate and test. You will create 3 such programs (but you only need to submit one). The first should solve the problem for only customers `c1`, `c2`, `c3`, `c4`, `c5`. The second should solve the problem for the first 8 customers (`c1` to `c8`). The third should solve the problem for the first 9 customers (`c1` to `c9`).

You should also submit your program for the 8 customer problem in the file `q1_8cust_generate_and_test.pl`. Please follow the format indicated in that file, including having comments briefly explaining the meaning of your variables. Your program should take the form of `solve(L)`, where `L` is the list of variables that the program will set. You should also complete the `printSolution` predicate, which will output the answer in a human readable format. You have flexibility on the exact format, provided that it is clear to the grader what order the customers are visited in. Note, that we have given you a predicate called `solveAndPrint` which calls `solve`, outputs the runtime, and also calls `printSolution`. This is the main way we will interact with your program.

You should then document the output of your program and runtime for all three problems in your report file, `assignment3_report.txt`. Note, you do NOT need to submit the program for the 5 or 9 customer problem. You only need to create those program to document the output and the runtime.

b. [25 marks] The second task is to write program using interleaving generate and test. You will create three such programs (but you only need to submit one). The first should solve the problem for the first 9 customers (`c1` to `c2`). The second should solve the problem for the first 17 customers (`c1` to `c17`). The third should solve the problem for all 18 customers.

You should then document the output of your program and runtime for all three problems in `assignment3_report.txt`. In addition, write a few sentences in this report describing how the results compare to the standard generate and test approach. You should also submit your program for the 17 customer problem in the file `q1_17cust.pl`. Please follow the format indicated in that file. The format should be the same as in part (a). If you are reusing the KB or the helper predicates from part (a), please copy the relevant parts to `q1_17cust.pl`. This file MUST run when it is loaded on its own.

As in (a), you do NOT need to submit the program for the 9 or 18 customer problem. You only need to create those programs program to document your results.

[3]In my solution, I actually wrote a Python script that would output the entire Prolog program for whatever variant (problem size or generate and test vs interleaving) that I needed. You don't need to take this approach, but it is worth thinking about as a way to combine different tools (Python and Prolog) to solve a real-life industrial problem.

2 Network Puzzle [40 marks]

In the town of Mooseville, there are just six computer systems connected to the Internet, though each system may have many users on it. The six computer systems exchange files with the systems linked with them on the map below. System 1 on the map also exchanges files with systems outside of Mooseville, providing the town with its connection to the rest of the world. Thus, electronic mail travels from system to system along the links shown on the map:

```
(6) -- (5) -- (3) -- (2) -- (1) --> (Rest of the world)
               |
              (4)
```

```
Note: 4 has a link to 3, 5 has a link to 3, 5 does not have a direct link to 4.
The path from 6 to "rest of the world" is: 6, 5, 3, 2, 1, out.
```

Each of the computer systems has a different Internet address (`bananas.com`, `firstbank.com`, `netvue.com`, `pricenet.com`, `sysworld.com`, and `univmoose.edu`). Each system also has a different systems administrator. The six systems administrators include three women named Catarina, Lizzie, and Mona, and three men named Anthony, Daniel, and Jaime. Their last names (in no particular order) are Elby, Kim, Osborne, Tsuji, Wolverton, and Zickerman. Your Prolog program must match each computer system on the map with its Internet address and with the full name of its system administrator. You know the following facts are true:

1. Email between Lizzie and Osborne must pass through the system with the Internet address `pricenet.com`.
2. Mona's and Wolverton's systems exchange files directly. One of these systems' addresses is `netvue.com`
3. Email between Anthony and Jaime must pass through Elby's system (and possibly others).
4. Among the systems in town, Daniel's exchanges files directly only with the one whose address is `sysworld.com`
5. Email between Jaime and Ms. Tsuji must also pass through the system whose address is `univmoose.edu`
6. The system whose address is `bananas.com` has a woman systems administrator.
7. Email between Kim and the rest of the world must also pass through the system whose address is `firstbank.com`
8. Among the systems in town, Zickerman's exchanges files directly only with Catarina's and the system whose address is `netvue.com`. Zickerman's system does not have direct links with any other computer system in the town.

a. [**5 marks**] You have been provided with a KB containing the network topology using `link` predicates in the file `q2_network.pl`. You can now use these facts to write a program with the name `connected(Origin, Destination, Path)` which finds the list of systems connecting `Origin` to `Destination` in the network. You will find this predicate useful in part (b) to formulate your constraints.

As an example of the usage of this predicate, in the network above, `connected(5, 4, X)` should succeed with `X = [5, 3, 4]`, and `connected(5, out, [5, 3, 2, 1, out])` should succeed. If `Origin = Destination`, then `Path = [Origin]`. You can assume that `connected` will never be called with either `Origin` or `Destination` set as a variable. However, your program should be general to other network topologies, not specific to the network topology given in your program. Write your program in the `q2_connected` section of `q2_network.pl`.

HINT: This program should be quite similar to `canReach` on Assignment 2, just without `MaxDist`.

b. [35 marks] Write a program to solve the given CSP. You should implement `solve` and `printSolution`, as defined in Question 1. These should be implemented in the correct sections in `q2_network.pl`. You must use interleaving for this question.

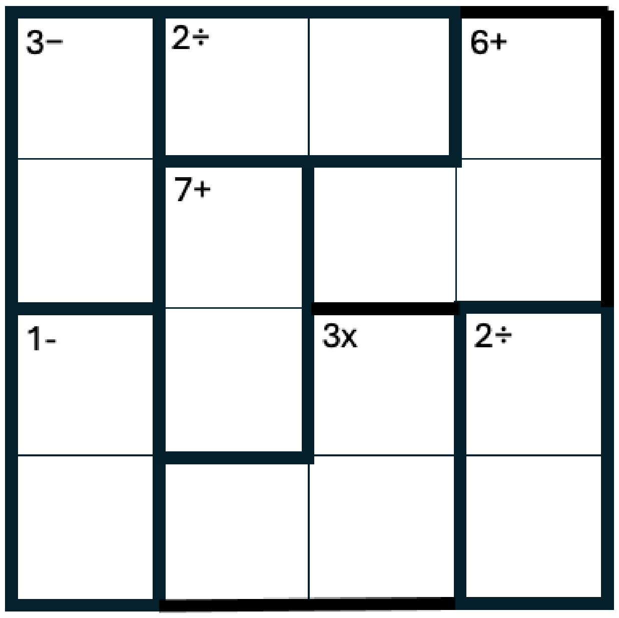
Please note the following:

- You can assume that in constraints like (1), the package may or may not pass systems other than `pricenet.com`.
- Constraints like (1) imply that Lizzie and Osborne are different administrators, and neither is the administrator of `pricenet.com`.
- When the clues say "exchange files directly" or there is a "direct connection" as in constraint (2), this means that Mona and Wolverton are administrators of systems that are beside each other (and consequently, they are not the same person).
- You can reorder the constraints — including any `allDiff` calls — as you see fit. However, do not combine clues (let the program do that reasoning itself), and clearly label in your program which lines correspond to which constraint.

In `assignment3_report.txt`, document the output of your program and how long it took to run.

3 KenKen [40 marks]

KenKen is a logic puzzle much like Sudoku, where the task is to fill grid cells with positive integers such that a number of specific constraints are satisfied. For this problem, you will be solving the following 4x4 puzzle:



The task is to put one of the integers from 1 to 4 in each of the 16 grid cells such that the following is satisfied:

- Each row contains exactly one of each integer from 1 to 4.
- Each column contains exactly one of each integer from 1 to 4.
- Each heavily-outline region is called a *cage*. Each cage must satisfy the arithmetic constraint shown in the upper left corner. For example, "5+" in the bottom left group means that the two digits in the corresponding cage must add up to 5. The "2−" in the top left group means that the difference between the two values is 2 (ie. 4 and 2 or 3 and 1). Note that the order of the numbers in the cage does not matter, so having 4 above 2 or 2 above 4 will both satisfy the constraint "2−". produce 2. The same is true of the divisor constraint "2÷".
- The integers within a cage can be the same or different, as long as they satisfy the arithmetic constraint.

Write a program to solve the given KenKen problem. You should implement `solve` and `printSolution`, as defined in Question 1. These should be implemented in the correct sections in `q3_kenken.pl`. You must use interleaving for this question.

HINT: you may want to define helper predicates for *subtract* and *divide*, which will succeed regardless of the order of the input arguments. However, be careful about colliding with the name of the built-in `subtract` predicate (*ie.* call such predicate something else). In addition, be aware that division using / returns a floating point number, not an integer (even if the inputs are integers).

In `assignment3_report.txt`, document the output of your program and how long it took to run.