Part 1 and Part 2 of Assignment Advanced Topics in Computer Science (2018/2018),

version from 28-2-19

Deadlines

The deadlines are firm and you cannot resubmit parts to improve your grade later.

- **23:00, April 7th, 2019: Part 1**
- **23:59, May 5th, 2019: Part 2**

Note that the description and the requirements of Part 3 will follow later in a different document at the due time.

Evaluation

Models are evaluated based on:

1) Technical requirements, i.e. does the model correctly implement the specification? Are there no deadlocks? Are there no tokens left behind? Etc. All these aspects will be addressed in the lectures.
2) Readability and understandability by a human. An essential aspect of modeling is to capture a complex situation with a model that is as readable and comprehensible as possible.

To evaluate readability, each model and report will be analyzed individually. In order to get intermediate feedback on the readability of your model, make use of the instructions.

This assignment is to be solved **in a group composed by two students.** While it is encouraged to discuss with students of other groups, the solutions have to be **original**. Solutions of the two or more groups that are almost identical will be considered as plagiarism ("copiato"), all those groups will be just given 0 points, with no possibility to repeat the assignment.

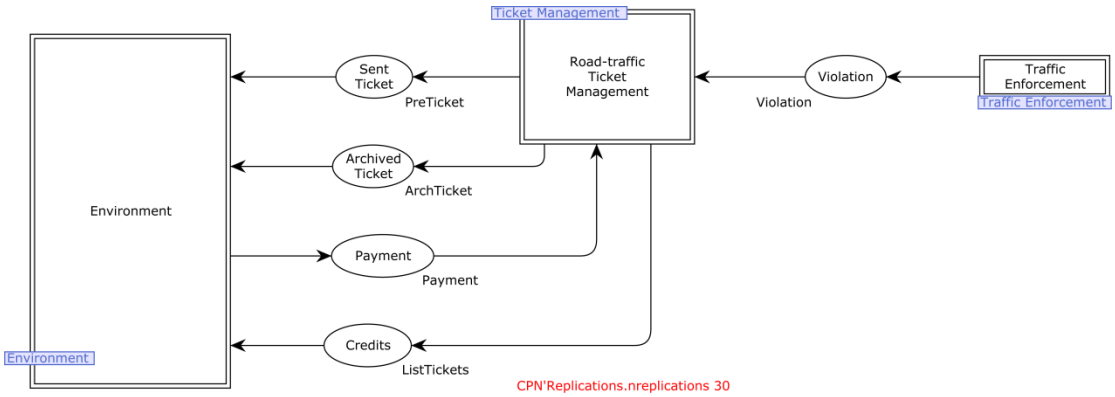
Part I and Part III will be each given a grade between 0 and 40; Part II will be given a grade between 0 and 20. The maximum sum will be 100, which will be later multiplied by 0.3 to scale between 0 and 30 and rounded to the closest integer. If the sum is greater than or equal to 99 (i.e. the average is greater than 29.7), this will lead to "cum laude".

Task and requirement

In the CPN assignment, you have to model how road-traffic tickets are managed by municipalities. For modelling, you need to use CPN Tools (version 4.0.1) available from http://cpntools.org. For every task, a part of the model is already given as a CPN. This model is provided as a file "**`baseModel.cpn`**" via Moodle.

Goal of the Assignment and Description of the Provided Part of the Model

The setting of the assignment is the office for managing road-traffic tickets of a small municipality.
Part of the model is already given in CPN Tools. The top level model is shown below.



The *Traffic Enforcement* and *Environment* subnets are already given. The former models the observation of violations of road-traffic laws. The *Environment* models the behavior of the "world" outside the office for managing road-traffic tickets. The *Road-traffic Ticket Management* subnet models the process to deal with the tickets associated with violations of road-traffic rules. It is initially empty and the purpose of the assignment is to complete it. The data is exchanged between the net modelling the road-traffic management and the other two through a number of places as the above figure shows.
The purpose and the use of expression "`CPN'Replications.nreplications 30`" is to analyze the flow time, which is required by the second part of the assignment. The last section of this document *Analyzing Flow Times* discusses how to use this expression by such an analysis.

Provided Color sets and Variables

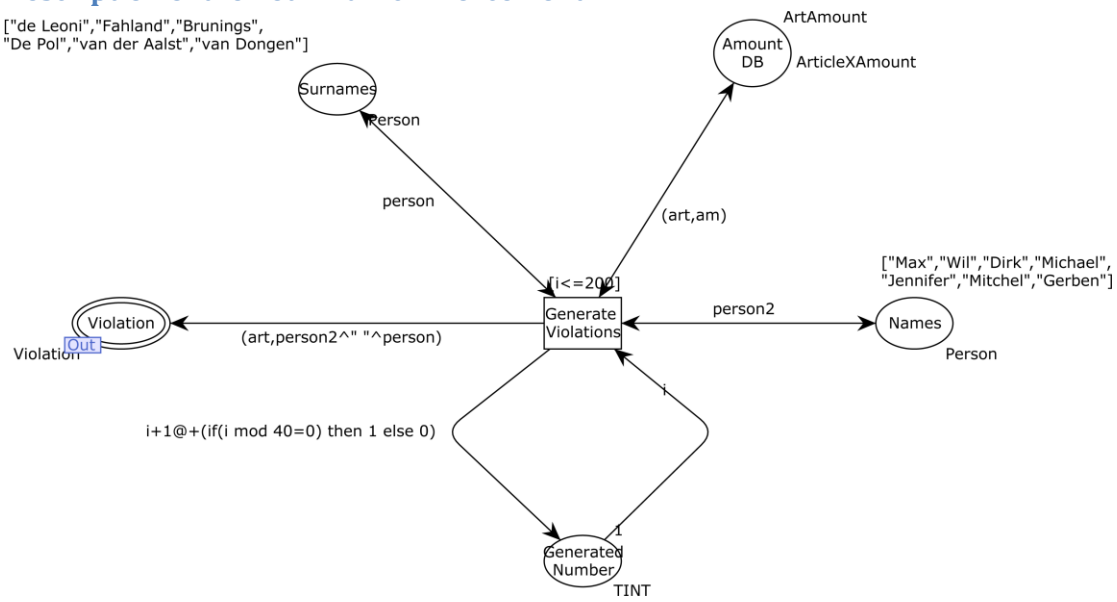
The following color sets and variables are given:

```
Initial Declarations
val ArtAmount=[(1,100),(2,50),(3,150),(4,500),(5,100),(6,75)];
colset Amount = INT;
colset PaidAmount = Amount;
var i :INT;
colset Article = INT;
colset Person = STRING;
colset TicketNum = INT timed;
colset TINT= INT timed;
colset LPaidAmounts = list PaidAmount;
colset Violation = product Article * Person timed;
colset ArchiveReason = with PAID | CCOLL | EXPIRED;
colset Ticket = product TicketNum * Amount * LPaidAmounts * Article * Person  timed;
colset Payment = product TicketNum * Amount timed;
colset PreTicket = product TicketNum * Amount * Article * Person timed;
colset ArticleXAmount = product Article * Amount;
colset ListTickets = list Ticket timed;
colset ArchTicket = product Ticket * ArchiveReason timed;
var n : TicketNum;
var am,am2: Amount;
var person,person2 : Person;
var lTicket,lTicket2 : ListTickets;
var lPay : LPaidAmounts;
var art : Article;
fun amountPaid([]) : Amount = 0 |
  amountPaid(am::lPay) : Amount =am+amountPaid(lPay);
```

In particular, it is worthy paying attention to the function `amountPaid`. Given a list `lPay` of payment, the function returns the total amount computed as the sum of the amounts of all payments in `lPay`. The function is already given and fully functional.

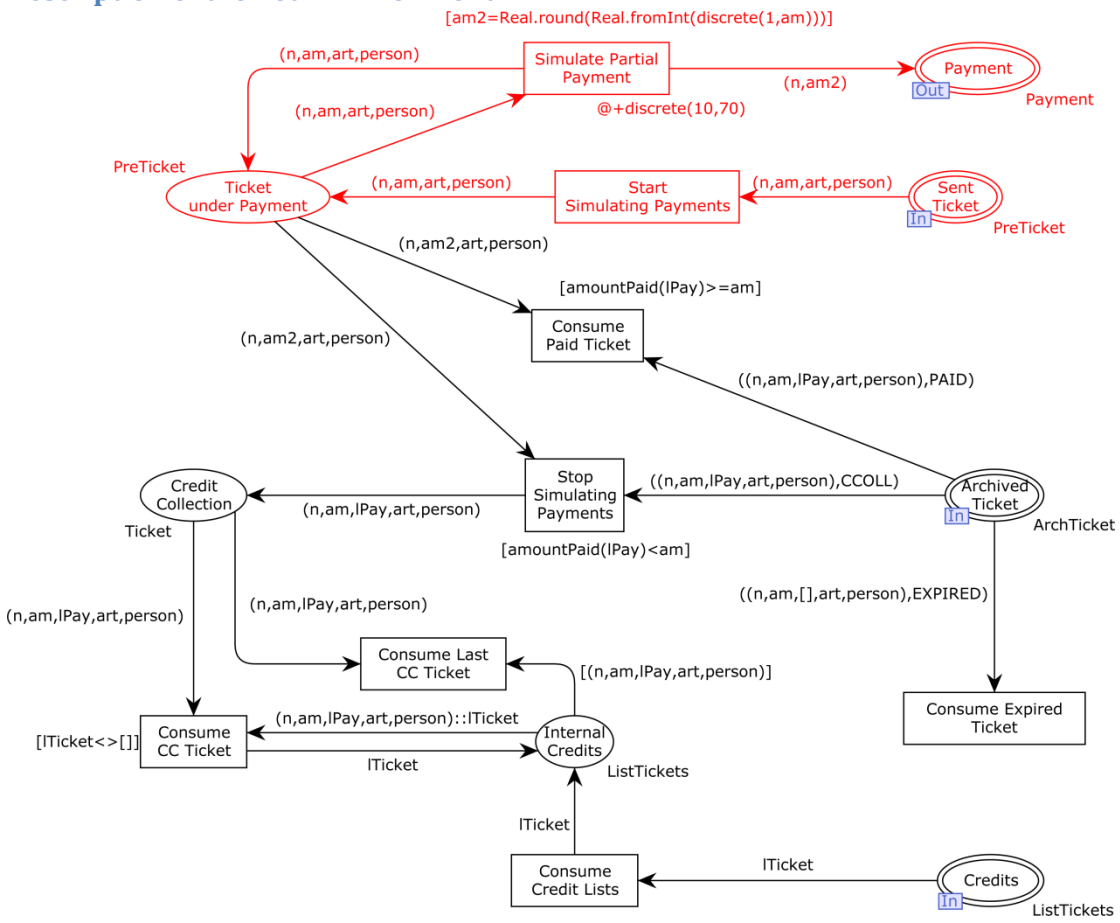
Please note that the provided definitions and function `amountPaid` cannot be changed in any case. Also see section *Rules of the game* at the end of this document.

Description of the net "Traffic Enforcement"



This net simulates the generation of violations by producing tokens for place *Violation*. Each token is a pair consisting of the violated law's article and the person committing the violation. Names are randomly generated by concatenating a name and surname, each taken from a predefined list. Articles are taken from the list kept in the value *ArtAmount*. This net simulates a 5-day scenario where 40 violations are committed per day.

Description of the net "Environment"



As mentioned, the net *Environment* is in charge of simulating the part of the "world" of interest.
The **Red part** simulates the payment of tickets by offenders. For each payable ticket, one payment is periodically generated with frequency uniformly distributed between 10 and 70 days until the ticket is archived (see requirements of Part 1).
The **Black part** consumes the tokens that are generated by the *Road-traffic Ticket Management* subnet.

This net can also be used to **partly** verify the correctness of the implementation of the *Road-traffic Ticket Management* subnet. If, at the end of a simulation, any token is left in any place, then the *Road-traffic Ticket Management* subnet is not implemented correctly. *However, please note that this is a necessary condition: it is possible that the subnet is incorrect even if no tokens are left in any place of the black part.*

Part 1 – Basic Model (40% of the grade)

Part 1 requires to

- build a CPN model that describes the process of managing road-traffic tickets and adheres to the requirements below;
- provide a report that discusses the solution proposed for the CPN model;
- provide a simulation of the model and attach the results of the simulation to the report. The analysis should be based on running at least 30 subruns where each run corresponds to a 5-day scenario where 40 violations are committed per day. This is the default setting; see also section *Analyze Models* at the end of this document.

Requirements

The management of a ticket starts when a violation is committed by an offender, which corresponds to the generation of a token in place *Violation*.

The following list describes the main requirements of the road-traffic ticket management:

1. When a violation is committed, the *Insert Ticket* activity is performed. This activity finds the amount associated with the violation in the database, here represented as the list stored in value *ArtAmount*, and binds it to the violation. Also, a ticket number is bound. The ticket number is progressive starting from 1. For instance, if the last ticket number was 1, the new bound number is 2, and so on. Inserting a ticket takes 1 day.
2. After inserting the ticket into the system, the ticket is prepared and sent by post to the offender. This corresponds to producing a token in place *Sent Ticket*. Preparing and Sending a ticket takes 1 day, as well.
3. After preparing and sending the ticket, the management office is awaiting for receiving payments. Each payment corresponds to the production of a token in place *Payment*. When the payment is received, it is added to the list of payments for that ticket. Processing the payment takes 2 days.
4. When the ticket is fully paid, the ticket is closed and also archived with reason PAID.
5. After 180 days, if the offender has not paid the amount due for their ticket in full, the ticket is prepared for credit collection and added to a corresponding list. Also the ticket is archived with reason CCOLL (Credit Collection).
6. Every 30 days, all tickets prepared for credit collection are sent out, which corresponds to producing a token for place *Credits*. Please note that, if no tickets are prepared for credit collection, no token is produced for place *Credits* until one ticket becomes available. The 30-day timer starts as soon as the previous round is sent out.
7. It may happen that payments are received after the ticket is archived (for credit collection or for full payment). These payments need to be refused. Processing a payment and refusing it takes 2 days. To keep simple, you do not need to model the fact that ticket offenders are notified about the refusal.
8. Assume that 15 employees are present for dealing with the incoming tickets. Only the performance of the following tasks require employees:
    - Inserting a ticket (see point 1)
    - Preparing and sending a ticket (point 2)
    - Preparing for credit collection (point 5).

Each employee can indistinctly perform any of the tasks above. The other tasks are automatically performed by the system without requiring the involvement of resources.

9. If the ticket is not sent within 10 days after being inserted, it expires, meaning that the offender does not have to pay the amount related to the violation. This means that the government of the town loses this amount. If the ticket expires, it is archived. When the ticket is archived, a token is generated for place *Archived Tickets* with the corresponding archiving reason (see colorset *ArchiveReason*): EXPIRED.

Your solutions should be *generic*, i.e., it should be easy to add new articles with the corresponding amount. In other words: the solution should be configurable and not depend on the particular set of available articles for violations with the respective amounts.

Important note**: The above process description is not intended to be a step-by-step description of the process, but rather some abstract requirements. If you feel that anything is underspecified, **you must resolve this underspecification** and come up with a concrete model satisfying all requirements. **The purpose of the instructions is that you ask about your ideas and assumptions as you come up with a solution.

The structure of the report for Part 1

In a report briefly describe the model and your design choices. Include a diagram of the corresponding CPN(s) that is as readable as possible. If necessary for increasing the readability, you can also use different colors for the arcs. The purpose of the report is to relate your technical solutions in the CPN model to the informal requirements. For this, your report should discuss every assumption that you have made which is not explicitly stated in the text. Also you should explain the transitions, color set definitions, arc inscriptions and guards that are not trivial (i.e., anything that is more than a simple tuple of variables). Ideally, the report is between 3 and 5 pages, including textual description and model.

How to submit the assignment for Part 1

- Provide the model for Part 1 as a CPN file named "**`ID[studentID]-part1.cpn`**" where studentID is the student identifier of one of the two members of the group (i.e., your file should be called "`ID0463033-part1.cpn`", if the student id of one of two member is "`0463033`").
- Provide a report, explaining the model of Part 1, as a pdf file named "**`ID[studentID]-part1.pdf`**" (i.e., your file should be called "`ID0463033-part1.pdf`", if the student id of one of two member is "`0463033`"). Ensure that the report clear states the name and student id of the group members.

Part 2 – Sensitivity analysis via simulation (20% of the grade)

As mentioned in the description of the requirements for Part 1, if a ticket is not sent within 10 days after being inserted into the system, the ticket expires. When tickets expire, the municipality loses the money amount associated with them.
In particular, the municipality is willing to hire new employees if this is convenient.
Hiring new employees will likely lead to a reduction of the number of expired tickets and the amount of money that is lost. Employees are paid as function of the number of tickets that are overall inserted into the system: one employee costs 2 Euros per ticket.
Through simulation, you need to understand if and until which point it may be convenient to hire new employees.

The analysis should be based on running at least 30 subruns where each run corresponds to a 5-day scenario where 40 violations are committed per day. This is the default setting; see also section *Analyze Models* at the end of this document. Note that, with this setting, each additional employee costs 2 Euros x 5 days x 40 violations = 400 Euros.

Of course, if the simulated model contains mistakes, the results are (partly) invalidated. Before running the simulation, ensure that you have fixed all mistakes of the model that you submit for part 1. Use the teacher's feedback on the solution on part 1 to fix; also use the instruction and walk-in session for this purpose.

The structure of the report for Part 2

In a report, briefly describe any changes with respect to the model of Part 1. Do not iterate again about what has not changed. Include a diagram of the corresponding CPN that is as readable as possible. If necessary for increasing the readability, you can also use different colors for the arcs. The purpose of the report for part 2 is to show:

- the methodology that you used to conduct the simulation;
- the results that you obtained; to illustrate more effectively, you can use graphs (bar charts, pie chart, etc.).

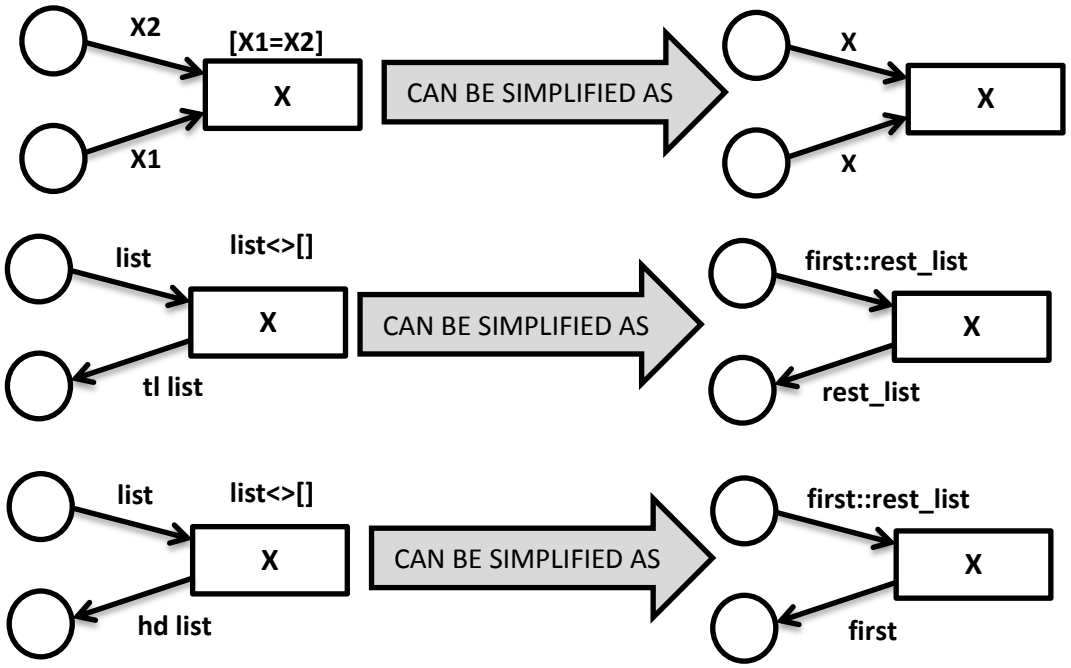
Ideally, the report is around 3-4 pages, including the possible graphs and the model diagram.

How to submit the assignment for Part 2

- Provide the model for Part 2 as a CPN file named "**`ID[studentID]-part2.cpn`**" where studentID is the student identifier of one of the two members of the group (i.e., your file should be called "`ID0463033-part1.cpn`", if the student id of one of two member is "`0463033`").
- Provide a report, explaining the model of Part 2, as a pdf file named "**`ID[studentID]-part2.pdf`**" (i.e., your file should be called "`ID0463033-part1.pdf`", if the student id of one of two member is "`0463033`"). Ensure that the report clear states the name and student id of the group members.

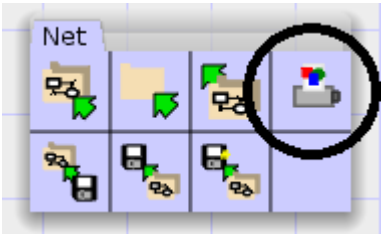
Rules of the game

- Always use the base file specific for you, do not use base files specific to other students.
- Do not change the base model; only add new functionalities. You must not change the given declarations. We recommend that you use an extra declaration block, for instance the provided block called "`Own Declarations`", for the declarations you add.
- Models in which the given declarations, net *Environment* and/or net *Traffic Enforcement* have been changed will not be judged.
- Models that have been uploaded after the deadline will not be judged.
- The grade for each part is determined directly after the respective deadline. There is no option to improve on the submission later.
- The final grade is determined after the deadline.
- This assignment is to be solved in a group composed by two students. While it is encouraged to discuss with students of other groups, the solutions have to be original. Solutions of the two or more groups that are almost identical will be considered as plagiarism ("copiato"), all those groups will be just given 0 points, with no possibility to repeat the assignment.
- Models will be judged on correctness (i.e., conformance to specification) and readability. Readability concerns among others the layout of the net structure (i.e., avoid a spaghetti-like net structure), the labeling of places, transitions, and functions (i.e., do not use cryptic names but use self-explanatory, understandable names) and the model complexity (i.e., data structures should not be unnecessarily complicated; it should be possible for other people to understand your model). Make use of the instructions to get an indication whether your models are readable or not. You can use colors for arcs and places to improve the readability of the model.
- For all solutions, you will receive feedback regarding correctness and readability of your models. Use this feedback to improve your submissions for later deadlines.



Hints

- Check the CPN Tools Web page (http://cpntools.org/) if you have questions about the CPN syntax and how to model with CPN Tools.
- If you want to take a picture of your model, you can use print feature available in the "Net tool box" and click on the button as shown below:



This will generate an EPS file, which you can convert into the appropriate format to add to your reports.

- Running a single simulation of a CPN model on a modern computer takes 1-2 minutes. If the simulation takes several minutes, then this is a hint that your model is too complex (e.g., you have places containing thousands of tokens) or the simulation does not even stop (i.e., in the `replication_report.txt` file in `output/rep_x/` no simulation is reported as completed). In this case, simulate your model manually to figure out the problem.

- If you experience that your simulation of 30 sub-runs is incomplete, i.e., the model is not simulated until the end and some of the reservations of the 2000 customers are completed in each run, then there can be two reasons: your model has an error that prevents completion of every reservation, or the simulator is set to stop after a specific number of steps before reaching the reservations for all customers. To check which one is the case, open the “Simulation tool box” and click on the “fast-forward” simulation. The “maximum number of steps” to simulate is highlighted. Set it to a large number, e.g., 500000, and run the simulation once.

This will set the simulator to stop after 500000 steps (or when there is no transition enabled anymore). If you now run your simulation of 52 sub-runs you should get complete simulation results. If not, your model contains an error. Use manual simulation/fast-forward simulation of 50 steps to find the issue.

Analyzing Models

CPN Tools allows one to automatically simulate your model and to collect statistics about the simulation in a report. Net *Environment* uses so-called monitors to measure certain statistics. In particular, three monitors are defined: Expired Tickets, Paid Tickets, Credit Collection Tickets. Respectively, they collect statistics about the number and the sum of the money amounts of the tickets in the three categories.
The statistics are stored in the subdirectory "Output" if a single run is simulated. If multiple subruns are generated, these are stored in subdirectory "Reps_X". For the assignment it is necessary to compute confidence intervals, so multiple subruns are needed. To do so, right-click on "`CPN'Replications.nreplications 30`" in the main net and, then, select "Evaluate ML" from the menu to start the simulation. See http://cpntools.org for details.

In order to properly collect the statistics, your CPN model should have the following options selected. By default, these options should already be set. In case you obtain empty simulation reports, please check these options first.