Author: Rein Spanjer, Bob, Jorrit Stutterheim
Group number: 10
Student ID Jorrit: 13957899
Student ID Rein: 13558307
Student ID Bob: 14076357
Study: Premaster Software Engineering
Course: Programmeertalen

Declarative characteristics

Question 1

Describe the parts of the eFLINT language used for defining relations (1 above), manipulating relations (2) and performing inference (3). Do not only list the constructs; describe how they are used to program with relations.

Relations in eFLINT are defined using the following key words:

- Fact: specifies an identity that has relations with other identities.
- Placeholder: does not directly define a relation, but is used later to define complex relations.
- Duty: Identifies a relation between two identities in which there is one party responsible for the duty and another party a claimant for that duty.
- Act: Defines a set of actions between 2 parties that create responsibilities for both parties.
- Identified by: Can describe a relation between types or just the type

```
Fact person Identified String
Placeholder parent    For person
Placeholder child     For person

Fact natural-parent   Identified by parent * child
Fact legal-parent     Identified by parent * child
  Holds when natural-parent(parent,child)

Domain:
Fact person Identified by Alice, Bob, Chloe, David, Ellis
```

The person is of the type String and the natural-parent is a relation between the parent type and the child type. The relation of the legal-parent is automatically given when the natural-parent relation is added.

Once the possible relation are defined it can be added in the initial state, like

```
natural-parent(Alice, Bob).
```

This will add a natural-parent and an legal-parent relation to Alice and Bob. To further manipulate these relations we can use Act and Duty. With Act we can introduce Actions that have taken place which could

trigger other things in the model.

```
Act ask-for-help
  Actor      child
  Recipient  parent
  Creates    help-with-homework(parent,child)
  Holds when legal-parent(parent,child)
```

The parent will only help the child if it asks for it and if it is a legal-parent.

Question 2

In eFLINT you can write ?., i.e. a question mark followed by a Boolean expression followed by a full stop. Using the terminology of mathematical relations, logic programming, or database programming, what is this construct used for?

In eFLINT this a boolean expression is used to validate if a Duty or Act has been performed according to the specifications.

A similar statement in logic:

```
p -> q
```

This construct is used to determine if the state is as expected. After some manipulations the programming can do this checks to be sure that the state is still as expected or that a error has been made. Even in application running in production this can monitor the state of the program. Sometimes you don't want to do something before it is checked. If the check fails somebody can look at the model and fix it before it becomes worse or unnoticed.

Question 3

What (non-syntactical) differences do you see between the language constructs discussed in the answers to Q1 and Q2 and similar constructs in Prolog or MySQL? Consider, for example, expressive power,or behavior when executed. You can also compare with other logic programming or database languages if you prefer. Be specific.

eFLINT looks like a language that is still in development. It lacks any support for syntax error messages. But the way that you can see all the different states of the program after each query is a nice feature to have. With MySql you have more freedom to request data and manipulate data. This has it downsides or upsides in comparison with eFLINT, it depends on the application. If I would like to have all the properties a records has I can do that in one easy query.

```
SELECT *
FROM PERSONS
WHERE AGE > 10
```

How would I do this in eFLINT?

Question 4

Describe the parts of the eFLINT language that can be classified as imperative instructions. Explain how you decided that these are indeed the instructions of the language. For each kind of instruction, explain the effects these instructions can have.

An imperative instruction is where you describe the flow of the program on how to solve something. In E-Flint a scenario can describe things that can happen. Events, observations or actions. These things are different in nature and can also be different in eFLINT. These things are introduced in eFLINT in a imperative manner.

Question 5

What would you say is the state manipulated by an eFLINT program? How does it change overtime? How does the manipulation of state help with assessing a normative (legal) case for compliance?

The state can be manipulated by doing an Act or by introducing facts like `+legal-parent(David,Chloe).` At every state it is clear what the Duty's are and if there are violations. In combinations with the validations after every state change it becomes immediately clear when there is a wrong state which is quite valuable in a legal case.

Semantic analysis

Question 6

Explain in detail when a duty is considered violated and in which ways a duty can become violated,for example the help-with-homework duty of the 'Help with Homework' example. In your explanation you can give and refer to examples. However, your explanation needs to be valid for all duties defined in the language.

A duty is considered violated when the duty is either not terminated before one of the violations gets true, or cannot be done because one of the violations is already true when the duty is made. In the example 'help-with-homework', the duty is not violated when it is made. In the next line, the fact is created that Bob's homework is due, which is in violation with the duty. The duty is terminated too late which causes the violation. If the fact 'homework-due' for Bob was stated before the duty was made, than the duty would be immediately in violation whenever the duty is made, unless the fact that Bob's homework is due has been terminated.

Question 7

Argue whether the language is statically or dynamically typed. How did you determine this?

The language is statically typed. In the frames section it is possible to omit types and even overwrite the types of the different Facts. But once this stage is successful and these types are used in the scenario these cannot change anymore.

Example:

```
-- Frames:
Fact myint Identified by Int

-- Scenario:
+myint(1).
+myint(2). // works without line below
+myint("hello world"). // outputs 'technical error'.
```

Question 8

Write a C function that implements the ask-for-help act from the 'Help with Homework'example. The goal is that when this function is called with some arguments, the function behaves like when an instance of the ask-for-help type is triggered. Alternatively: use Haskell, Prolog, MySQL or pseudo-code instead of C.

```
type Name = String
type Relation = Name
type ParentName = Name
type Child = (Name, ParentName)
type Holder = Name
type Claimant =  Name
type ChildName = Name
type Parent = (Name, ChildName)

isViolated :: (a -> Bool) -> a -> Bool
isViolated p x | p x  = True
               | otherwise = False

isLegalParent ::  Child -> Parent -> Bool
isLegalParent c p | fst c == snd p && fst p == snd c = True
                  | otherwise = False

createHomeworkDuty ::  Child -> Parent -> (Holder, Claimant, (a -> Bool) ->
a -> Bool)
createHomeworkDuty c p = (fst p, fst c, isViolated)

askForHelp :: Child -> Parent -> Maybe (Holder, Claimant, (a -> Bool) -> a
-> Bool)
askForHelp c p | isLegalParent c p = Just $ createHomeworkDuty c p
               | otherwise = Nothing
```