problem 1	:	25%
problem 2	:	25%
problem 3	:	25%
problem 4	:	20%
problem 5	:	5%
TOTAL	:	100%

All specifications, definitions, etc., which is part of your solution must be expressed using the specification language RSL.

Problem 1.

This problem concerns the specification of a library system. The system must store three kinds of information:

1. Which books the library owns. (It is assumed that the library does not own multiple copies of a book.)
2. The persons which are allowed to borrow books.
3. For each book borrowed, the borrower.

It is not further specified what books and persons are.

Question 1.1
Define a type: "Book" for representing books, a type "Person" for persons, and a type "Library" for library systems.

Hint Define "Library" as a subtype delimited by a well-formedness predicate which you also must define.

A variable, "lib" of type "Library" is now introduced. The functions in the following problems must all be imperative in the sense of reading and/or writing the variable "lib". Remember to state the necessary pre-conditions.

Question 1.2
Write an explicit definition of an imperative function, "owns", which checks whether a given book is owned by the library.

Question 1.3
Write an explicit definition of an imperative function, "is_borrowed", which checks whether a given book is borrowed from the library.

Question 1.4
Write an explicit definition of an imperative function, "borrower", which checks whether a given person is allowed to borrow from the library.

Question 1.5
Write an explicit definition of an imperative function, "borrow_book", for registering that a given person has borrowed a given book.

Question 1.6
Write an implicit definition (pre/post) of an imperative function, "return_book", for registering that a given person has returned a given book.

Problem 2.

This problem concerns specification of digital picture representations. A digital picture consists of points in a matrix with n columns and m rows, where n and m are constants not further specified. Each point in the matrix has a unique colour.

Question 2.1
There are many possibilities for modelling colours. Write different definitions of a type, "Colour", corresponding to each of the following possibilities:

1. It is not further specified what a colour is.
2. There are only two colours: black and white.
3. There are 256 colours which are represented by integers in the interval from 0 to 255 (greyscale representation).
4. There are all the colours obtainable by mixing the three primary colours (red, green, blue), each with an intensity indicated by an integer in the interval from 0 to 255 (colour representation).

In the following you may assume that the type: "Colour" has been defined.

Question 2.2
Define the constants "n" and "m", and a type "Point" for representing points in the picture matrix.

Question 2.3
Define a type "Picture" of digital pictures.

Question 2.4
Define a function "colour_of", which yields the colour of a given point in a given picture.

Question 2.5
Define a function "one_colour", which checks whether a given picture has only one colour.

Question 2.6
Define a function "vertical_stripe", which checks whether a given picture has a vertical stripe, i.e., a one-coloured bar whose neighboring columns have different colours.

Question 2.7
Assume that there is a function with the following signature:

$$+ : \text{Colour} \times \text{Colour} \rightarrow \text{Colour}$$

which can add two colours. The function is not further specified.

The sum of two digital pictures $p1$ and $p2$ is a new picture p where the colour of each point x is the sum of the colour of x in $p1$ and the colour of x in $p2$.

Write an axiomatic definition of a function, "+", for adding two digital pictures.

Problem 3.

This problem concerns the specification of finite, binary relations between the values in two possibly different types, which we will call the domain type and the range type. The values in the domain type are called domain elements and the values in the range type are called range elements.

Such a relation, r, consists of a finite collection of pairs of domain and range elements. We say that these pairs are in the relation and for each such pair we say that the elements of the pair are related in r.

Question 3.1
Write an algebraic specification of finite, binary relations. The specification must consist of a scheme, "RELATION", which is parameterized by two objects, "D" and "R", for specifying the type of domain and range elements, respectively. The scheme must contain definitions of:

1. A type, "Relation", of relations.
2. A constant "empty" representing the empty relation, i.e., the relation in which no elements are related.
3. A function "add" for adding a pair to a given relation.
4. A function "is_in", which checks whether a given domain and a given range element are related in a given relation.
5. A function "equal" which checks whether two relations are equal.
6. A function "image", which gives the set of range elements which are related to a given domain element in a given relation.
7. A function "$\cup$" for making the union of two given relations, i.e., the set of all pairs from both relations.
8. A function "$\backslash$", which from a given relation removes all pairs with domain elements equal to a given domain element.

Hint The specification must fulfill the following requirement: All relations must be (finitely) generated by the constructors "empty" and "add".

Problem 4.

This problem concerns the specification of a command language for manipulating relations in a database.

Notice: In the solution to this problem you may make use of the functions from problem 3 even if you have not solved problem 3.

A *database* consists of uniquely named relations, where relations are as specified in problem 3. It is not further specified what relation names are.

In order to describe the syntax of commands, we will first describe the syntax of so-called "relation expressions" which can be constructed in the following five ways:

- "n" where n is a relation name.
- "empty"
- "add d, r to e" where d and r are domain and range elements, respectively, and e is a relation expression.
- "e_1 union e_2" where e_1 and e_2 are relation expressions.
- "e minus d' " where d is a domain element, and e is a relation expression.

A relation expression may contain relation names, and its semantics is therefore defined with reference to a database including those relation names. The semantics of a relation name is the corresponding relation in the database considered. The semantics of the four other kinds of expressions are given by application of the "empty", "add", "$\cup$", and "$\setminus$" functions from problem 3, respectively.

A *command* is either a create, replace, or delete command. Commands are evaluated in the context of a database and yields as result a new database:

- "create n as r" extends the database with the relation denoted by the relation expression r, named n (n must not already be in use).
- "replace n by r" is like a create command, except that the name n must identify an existing relation in the database which is to be replaced by the relation denoted by the relation expression r.
- "delete n" deletes an an existing relation in the database.

Question 4.1
Define a scheme "LANGUAGE", which is an extension of the scheme "RELATION" from problem 3, including definitions of:

1. A type "Name" of relation names.
2. A type "Db" of databases. The definition must be model-oriented, i.e., it must use the built-in RSL type constructors.
3. A type "Command" and a type "Expr" for representing the abstract syntax of commands and relation expressions, respectively.
4. Two functions both named "is_wf" for checking the well-formedness of commands and relation expressions, respectively, in the context of a database. In other words, specify a suitable static semantics.
5. Two functions both named "M" specifying the meaning (semantics) of commands and relation expressions, respectively, by use of the functions defined in problem 3.

Problem 5.

This problem concerns the specification of a process, "result", which inputs three votes from three channels, one from each channel. Thereafter it outputs the result of the vote on a fourth channel. A vote is one of the values **true** and **false**. The vote is by majority.

Question 5.1
Define a scheme "VOTE" which contains definitions of:

1. the process "result", and
2. the four channels