Homework

Note

Write your answers on a plain text file and submit it via email (hlin@iis.sinica.edu.tw) with "[your_name,student_ID]" contained in the subject. You are encouraged to confirm your answer with the Overture tool. You can write your answers in English or Chinese. In the Overture tool, Chinese characters (kanji) can be used as names of types, functions, etc. in a VDM-SL model.

1 Logical Expressions

Translate the following statements into logical expressions: (use VDM-SL expression with quantifiers)

1. All the numbers in the set {7, 55, 133, 200} are greater than 5.
2. There is a Natural number less than 50.
3. There are two Natural numbers that multiply together to give 60.
4. The number 24 is even.
5. For any two successive Natural numbers, one of them is even.
6. There is no even number whose successor is even.
7. For any pair of natural numbers, there is a rational number equal to their quotient. (This isn't true: the divisor must not be zero. Rewrite the expression to record this restriction.)

2 Train reservations

In the UK the train reservation system works in a way where a small reservation tag is placed at reserved seats. This tag is simply a piece of paper so when you get to the seat to find someone in it and no reservation tag. You end up stading all the way to Edinburgh or wherever. Danish railways get over the problem by having a software reservations system that sends reservation data to the train where it gets displayed in a little LCD panel over the seat. This way, you arent subject to the railways staff making a mistake with reservation tags or with dodgy individuals removing the seat reservation when they get ito it. Suppose you work for a software firm commissioned to develop a booking system

on the Danish model. Your formal model refers to stations, trains and journeys. The purpose of your model is to define the functionality of the system. You might not be too concerned about how precisely stations are represented:

```
Station = token;
```

A journey could be modelled as a sequence of stations that you go through:

```
Journey = seq of Station;
```

Reservations are for segments of a journey (e.g. between the second and fifth stations):

```
Segment :: origin       : nat1
           destignation : nat1
inv s == s.origin < s.destignation;
```

Trains will be identified by identifiers, the respresentation of which is immaterial:

```
TrainId = token;
```

For each train, we record its route (the journey it is following) and the collection of seat numbers available on the train:

```
TrainInfo :: route : Journey
             seats : set of SeatNo;
```

We will not be concerned with the details of seat numbers:

```
SeatNo = token;
```

Now to reservations. These are identified by reservation identifiers (ResId):

```
ResId = token;
```

Information about each reservation includes the seat number reserved, the train it's on and the segment of the journey for which the reservation has been made - it's possible to have a seat reserved for someone from station 3 to station 5 and then reserved for someone else from station 5 to station 7, for example. The overall reservations system is a record of two mappings containing train details and reservation details:

```
System :: trains : map TrainId to TrainInfo
          res    : map ResId to ResInfo
```

2.1 Invariant

Define the following clauses in the invariant:

1. All the trains in which reservations have been made are known to the system (i.e. are in the trains mapping)
2. All reserved seats are actually in the trains in which they are reserved. (You will need to define `ResInfo` according to above description)
3. There is no double booking (i.e. there are no two distinct reservations which have reserved the same seat on the same train for some part of the trains journey). Here you will need to define what it means for two journey segments to overlap one another: do this by means of an auxiliary function.

2.2 Functionality

A function is satisfiable if, for any inputs satisfying the precondition (if there is one), the output is guaranteed to satisfy the invariant on the output type. Define the following functions and explain how you have ensured that your function is satisfiable in each case: (You may define either implicit or explicit functions, or both.)

1. Define a function that initializes the system to an empty system (i.e. a record with two empty mappings in it.
2. Define a function to add a new train to the system (i.e. add a train which so far has no reservations).
3. Add a new reservation for a given seat on a given train.
4. Suggest a seat: given a train and a desired journey segment, suggest an available seat (i.e. one that is not booked for the segment).