Tutorial 4

1) Differentiate between *pre-conditions* and *post-conditions* in Z schema.

2) You have been assigned by your company to model a car parking system that will record information about cars that enter the car park or leave the car park. The car park has a limited number of spaces called *capacity*. When a car enters or leaves the car park, the record about the car will be updated accordingly.

Given the basic type for the system:

[CAR] - the set of all possible cars

a global constant, *capacity*, as the maximum number of spaces in the car park:

$$\text{capacity} : \mathbb{N}$$
$$\text{capacity} \leq 200$$

and a state space, *CarParkSystem*, that will specify the properties of the system:

CarParkSystem

$$cars : \mathbb{P}\ CAR$$
$$\#cars \leq capacity$$

Referring to the specification above, write suitable Z schemas to represent the following operations for the system:

(a) A schema called *InitCarPark* which defines the initial condition of the car park where the car park is empty.

InitCarPark

$$CarParkSystem$$
$$cars = \varnothing$$

(b) A schema called *EnterCarPark* which adds a car, *car?* to *cars* provided its capacity has not been reached.

EnterCarPark

Δ *CarParkSystem*
car? : *CAR*

---

car? $\notin$ *cars* (car? does not exist in the system)
#*cars* < *capacity* (the capacity has not been reached)
cars′ = *cars* $\cup$ {*car*?}

have update in database show delta symbol Δ

(c) A schema called *DepartCarPark* which removes a car, *car?* from *cars* provided there is at least one car in the car park.

DepartCarPark

Δ *CarParkSystem*
car? : *CAR*

---

car? $\in$ *cars* (car? must exist in the system)
cars′ = *cars* $\setminus$ {*car*?}

(d) A schema called *SpaceAvailable* which produces the number of available parking spaces, *space!* in the car park.

SpaceAvailable

Ξ *CarParkSystem*
space! : $\mathbb{N}$

---

space! = *capacity* − #*cars*

(e) A schema called *DisplayCars* which produces all the cars in the car park.

DisplayCars

Ξ *CarParkSystem*
allCars! : $\mathbb{P}$ *CAR*

---

allCars! = *cars*

(f) A schema called *DisplayCars* which produces the total number of car in the car park.

DisplayCars

$$\Xi CarParkSystem$$
$$allCars! : \mathbb{N}$$

$$allCars! = \#cars$$

3) A university provides a multi-user computer system for its students and staff. All staff and students must register with the university's IT Services unit before they are allowed to access the computer system. To use the system, each registered user must logged-in. At any given time, a registered user will either be logged-in or logged-out (it is not possible for a user to be logged-in more than once concurrently). A registered user can only logged-in if there is an available machine, as the university has a finite number of machines.

A partial draft Z specification of the proposed system is provided below:

- a basic type:
  [PERSON] - the set of all uniquely identifiable persons

- a global variable declaration:

$$capacity : \mathbb{N}$$

$$capacity \leq 100$$

- and a state space schema, *ComputerSystem*:

ComputerSystem

$$users : \mathbb{P}\ PERSON$$
$$loggedIn : \mathbb{P}\ PERSON$$

$$loggedIn \subseteq users$$
$$\#loggedIn \leq capacity$$

Referring to the specification above, write suitable Z schemas for the following operations:

(a) An operation schema to register a new user called *RegisterNewUser*.

RegisterNewUser

$\Delta ComputerSystem$
$users? : PERSON$

$user? \notin users$ (user? does not exist in the system)
$user? \notin loggedIn$ (not important)
$users' = users \cup \{user?\}$
$loggedIn' = loggedIn$ (remain unchanged)

(b) An operation schema used by an existing user to logged-in into the system called *LoggedIn*.

LoggedIn

$\Delta ComputerSystem$
$users? : PERSON$

$user? \in users$
$user? \notin loggedIn$
$\#loggedIn < capacity$
$users' = users$
$loggedIn' = loggedIn \cup \{user?\}$

(c) An operation schema to display the total number of users who are logged-in into the system called *DisplayLoggedIn*.

DisplayLoggedIn

$\Xi ComputerSystem$
$loggedIn! : N$

$loggedIn! = \#loggedIn$