SOEN331/W: Introduction to Formal Methods for Software Engineering

Assignment 2 on axiomatic specifications of University Sports Management System

In groups of **2** students

Weight: 10% of the overall grade.

Date posted: February 16th, 2017.

Date due: March 1st by midnight.

Submission instructions: submit electronically (EAS) as "Theory Assignment 2", one submission per team. List the names and the IDs of the team members on the first page of your submission.

Goal: to practice axiomatic formalization of requirements with Pre-, Post-Conditions and Invariants

Guidelines: Each requirement is described by two assertions (or conditions or predicates) – precondition and postcondition. The precondition for a requirement asserts what must be true before any function(s) or operation(s) implementing the requirement is(are) invoked. The postcondition asserts what must be true after the function(s) or operation(s) that implements the requirement terminates. Typically, we derive the pre and postconditions from the description of the problem, and not from the code. Therefore, one can start writing the formal specification of the requirements without committing to any particular design or implementation. It can be used to develop the design later or can be used to verify the design and implementation.

In this assignment you are required to specify formally requirements using the format given in the lecture notes and provided below:

<Name> (<input parameter(s)>) <output parameter(s)>
Precondition
Postcondition

Both Precondition and Postcondition have to be formalized in predicate logic. We use the notation ***p'*** to denote that the global variable ***p*** has been changed after an operation. Typically, variables of the form ***p'*** will appear only in the postconditions.

1. Problem Description:

In this section, we describe the requirements of University Sports Management System.
A computerized system maintains information about players and teams in several universities and games played between the universities. Each university has a set of players, considered to be the sportsmen of the university. The players for a team are selected from the set of players of the

university. A university may have more than one team whose players are selected from the same pool, and so a player from a university may play for more than one team.
The following rules must be obeyed by the universities and their teams:

• Each team is associated with exactly one university.

• Each player is associated with exactly one university.

• A university may have more than one team.

• The players for a team are chosen from the players of the same university.

• A player from a university may play for more than one team of the same university.

• No two teams from the same university play against each other.

Assumptions

1. Since the problem is concerned with the relationships between teams, players and games, no personal information about players will be included.

2. A team will have a team name and a set of players. No additional information such as captain or team logo will be included in the specification.

3. Team names are assumed to be unique throughout the system.

4. University names are assumed to be unique.

5. For a game, only the two team names playing in the game will be maintained in the system. No additional information such as game timings or location will be included.

1.1 Basic types

The following types are assumed to be basic types in the current specification:

Player
Team Name
University Name

By "basic", we mean that these type names are used throughout the specification but the internal representational details of these types are not defined in this document.

1.2 User defined types

```
Team = structure of
            name: Team Name
            players: set of Player
       end
University = structure of
            name: University Name
            teams: set of Team
            players-pool: set of Player
           end
```

Game = structure of
team1, team2: Team
end

1.3 Global variables

There is one global variable used in this document:

Games: sequence of Game

where *Games* represents a sequence of scheduled games between universities.

2. Invariants

The state invariant must assert the players of all teams within the university must be coming from the pool of players belonging to the university **Write the invariant formally.**

3. Formal Specification. Write axiomatic specifications for the following operations:

3.1 add player to the University

AddPlayerToUniversity (u : University; p : Player)

Precondition:

- *ensure that there is no player **p** in the players pool of the university **u**, and*
- *ensure that there is no player **p** in the teams of the university **u***

Postcondition:

- *ensure that after the operation is completed, there is player **p** in the players pool of the university **u**, and*
- *ensure that there is no player **p** in all teams of the university **u***

3.2 add a team

AddTeam (u : University; tn : Team Name)

Precondition:

- *ensure that there is no team **tn** at university **u***

Postcondition:

- *ensure that after the operation is completed, there is a team **tn** with no players in the university **u** that has a unique name, and*
- *ensure that the players pool of the university **u** did not change*

3.3 add a player to a team

AddPlayerToTeam (u : University; p : Player; tn : Team Name)

Precondition:

- *ensure that there is a team **tn** at the university **u***
- *ensure that player **p** is in the players pool of the university **u***
- *ensure that player **p** is not in the team **tn***

Postcondition:

- *ensure that after the operation is completed, there is player **p** in the team **tn**, and*
- *ensure that the players pool of the university **u** did not change*

3.4 delete a player from a team

DeletePlayer-FromTeam (u : University; p : Player; tn : Team Name)

Precondition:

- *ensure that there is a team* ***tn*** *at the university* ***u***
- *ensure that player* ***p*** *is in the team* ***tn***
- *ensure that player* ***p*** *is in the players pool of the university* ***u***

Postcondition:

- *ensure that after the operation is completed, there is no player* ***p*** *in the team* ***tn****, and*
- *ensure that the players pool of the university* ***u*** *did not change*

3.5 delete a team

DeleteTeam (u : University; tn : Team Name)

Precondition:

- *ensure that there is team* ***tn*** *at the university* ***u***

Postcondition:

- *ensure that after the operation is completed, there is no team* ***tn*** *at the university* ***u****, and*
- *ensure that the players pool of the university* ***u*** *did not change*

3.6 delete a player from university

This operation needs special attention because a player when deleted from a university must also be deleted from all the teams in which he/she participates.

DeletePlayerFromUniversity(u : University; p : Player)

Precondition:

- *ensure that player* ***p*** *is in the players pool of the university* ***u***

Postcondition:

- *ensure that after the operation is completed, no team at the university* ***u*** *has a player* ***p***
- *ensure that there is no player* ***p*** *in the players pool of the university* ***u***

3.7 add game

AddGame(team1, team2 : Team Name)

Precondition:

- *ensure that these two teams must belong to two different universities*

Postcondition:

- *ensure that there is a new entry in the sequence of Games corresponding to team1 and team2*