Informatics Institute of Technology
Department of Computing
Assignment Cover Sheet

Course: **BEng (Hons) Software Engineering**

Unit Code and Description: ECSE610 Formal Specification

Module Leader: Dr. Sanjeewa Perera (ssnp@maths.cmb.ac.lk)

Assignment: Coursework 1 **Assignment Type: Individual**

Assignment Description: Develop a Z Specification of a Student Club

Hand – in – Date: 08/12/2014 **Time: 8:30 AM**

Student Name	**Yehan Chathuranga Pemarathne**
Student ID	**2011073**

The department is not responsible if an assignment is lost. To cover this eventuality you are advised to take a photocopy of the assignment OR to ensure you have the means of re-creating it.

1. Procedure for Handling Work:
   - Follow any specific instructions given on the assignment specification.
   - All written work should be placed in the box provided by the Registrar's Department on or before the date indicated on the cover sheet.
2. Penalties for Late Hand In:
   - If students submit coursework late but within 24 hours (or one working day) of the specified deadline, the work will be marked and will then have 10% of the overall available marks deducted, to a minimum of the pass mark (40% at Undergraduate level, 50% at Postgraduate level).
   - If students submit coursework more than 24 hours (or one working day) after the specified deadline, they will be given a mark of zero for the work in question.
3. Exceptional Factors Affecting your Performance:
   - Students should submit written evidence to the Registrar's Department with a copy to the Module Tutor of exceptional circumstances, which they consider to have caused them to submit assessments late and for which they do not wish to attract any penalty. These have to be handed over to the Registrar within four working days of the hand-in-date.
4. Assessment Criteria
   As indicated in the coursework.

Office Use Only (Registrar Date Seal):

1. Error correction in initial Club Specification

(corrections.txt)

Error 00
========
In the initial Club specification in the tute, semicolons are not there to separate variable definitions or invariants. Because plain specification doesn't need those. But it is obvious to have semicolons to separate variable definitions and invariants in ZBX form.

In the initial Club specification in the tute, there are no empty lines between type definitions. Because plain specification doesn't need those. But it is obvious to have empty lines to separate type definitions in ZBX form.

So these things are fixed at the initial writing of ZBX specification. Thus not included in errors or corrections.

Error 01
========
Have used wrong syntax "subeq" at two points.

Correction:
Changed both "subeq" to "subseteq"

Fixed:
```
--- Syntax error. "club.zbx" Line 22, near ";"
      Expecting: "=" "in" (infix relation symbol) "_inrel-id_"
>>>   committee subeq members ;
--- Syntax error. "club.zbx" Line 26, near "    ---------------------------
----------
"
      Expecting: "=" "in" (infix relation symbol) "_inrel-id_"
>>>         -------------------------------------
```

Error 02
========
Spelling mistake in Basic Type "[STUDENTS]". It must be "STUDENT" without taling 'S'. Because of this, number of errors occurred.

Correction:
Changed "[STUDENTS]" to "[STUDENT]".

Fixed:
```
--- Typing error. "club.zbx" Line 17. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 17. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 17. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 18. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 18. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 18. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 19. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 19. Illegal type expression:
>>>   STUDENT
--- Typing error. "club.zbx" Line 22. Type mismatch: Infix relation.
```

```
>>>   committee subseteq members
      Expected LHS type: (P [X])
      Actual LHS type: UnknownType
      Expected RHS type: (P [X])
      Actual RHS type: UnknownType
--- Typing error. "club.zbx" Line 28. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 28. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 28. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 29. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 29. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 29. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 30. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 30. Illegal type expression:
>>>   STUDENT
--- Typing error. "club.zbx" Line 33. Type mismatch: Infix relation.
>>>   newmember? notin members
      Expected LHS type: [X]
      Actual LHS type: UnknownType
      Expected RHS type: (P [X])
      Actual RHS type: UnknownType
--- Typing error. "club.zbx" Line 36. Type mismatch:
      LHS and RHS must have matching types.
>>>   committee' = committee
      LHS type: UnknownType
      RHS type: UnknownType
```

Error 03
========
Type mismatch of LHS and RHS of a system invariant at "committee <= MaximumCommitteeSize".

Correction:
Used cardinality of committee. So the correction is "# committee <= MaximumCommitteeSize"

Fixed:

```
--- Typing error. "club.zbx" Line 23. Type mismatch: Infix relation.
>>>   committee <= MaximumCommitteeSize
      Expected LHS type: Z
      Actual LHS type: (P STUDENT)
      Expected RHS type: Z
      Actual RHS type: Z
```

Error 04
========
"president" has defined as a subset of "committee" set which is not true. "president" is an element of "committee" set.

Correction:
Changed "president subseteq committee" to "president in committee".

Fixed:

```
--- Typing error. "club.zbx" Line 24. Type mismatch: Infix relation.
>>>   president subseteq committee
      Expected LHS type: (P [X])
      Actual LHS type: STUDENT
      Expected RHS type: (P [X])
```

```
      Actual RHS type: (P STUDENT)
```

Error 05
========
Type mismatch of LHS and RHS of a system invariant at "members < MaximumClubSize".

Correction:
Used cardinality of members. So the correction is "# members < MaximumClubSize"

Fixed:

```
--- Typing error. "club.zbx" Line 34. Type mismatch: Infix relation.
>>>   members < MaximumClubSize
      Expected LHS type: Z
      Actual LHS type: (P STUDENT)
      Expected RHS type: Z
      Actual RHS type: Z
```

Error 06
========
Type mismatch in "set union" at "members || newmember?" since the "newmember?" is not an element of power set of STUDENT as "members" is. "newmember?" is only an element of STUDENT.

Correction:
Changed "newmember?" to "{ newmember? }" so it becomes an element of power set of STUDENT.

Fixed:

```
--- Typing error. "club.zbx" Line 35. Type mismatch: Infix expression
>>>   members || newmember?
      Expected type of the infix expression: ((P [X]) x (P [X]))
--- Typing error. "club.zbx" Line 35. Type mismatch: Right-hand side
>>>   newmember?
      Expected type: (P STUDENT)
      Actual type: STUDENT
```

Error 07
========
Note: This error must have caught earlier this point. The error is in the original source of the Club specification. But I have initially typed it with the correction. But I have found out now. To be consistent with the coursework, I reintroduced the error and corrected it as follows.

In JoinClub_Success, there is a variable defined as "committee, committee : P STUDENT". One of these "committee"s must be "committee'". Because of this, an undefined variable error was also reported.

Correction:
Changed first "committee" variable to "committee'".

Fixed:

```
--- Typing error. "club.zbx" Line 29. Duplicate definition of name:
committee
--- Typing error. "club.zbx" Line 36. Undefined name: committee'
```

(corrections.txt)

Command Prompt

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Yehan>cd D:\Documents\IIT\4th Year\FS\FS-CW1\ztcwin

C:\Users\Yehan>d:

D:\Documents\IIT\4th Year\FS\FS-CW1\ztcwin>ZTC.EXE club.zbx
This is ZTC. Version 2.1.0a. Built on Apr 12 1996, 09:19:34 U.S.A. CST.
Copyright (c) Xiaoping Jia, 1993-1996.
... Initializing.
... Loading Z mathematical tools library: math0.zbx
Parsing main file: club.zbx
... Type checking Given set. "club.zbx" Line 4
... Type checking Free type definition: CLUB_NAME. "club.zbx" Line 6
... Type checking Axiom box. "club.zbx" Lines 8-11
... Type checking Free type definition: REPORT. "club.zbx" Line 13
--- Syntax error. "club.zbx" Line 22, near ";"
        Expecting: "=" "in" (infix relation symbol) "_inrel-id_"
>>>     committee subeq members ;
--- Syntax error. "club.zbx" Line 26, near "    ---------------------------------
-----
"
        Expecting: "=" "in" (infix relation symbol) "_inrel-id_"
>>>             ------------------------------------

... Type checking Schema box: Club. "club.zbx" Lines 15-23
--- Typing error. "club.zbx" Line 17. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 17. Set expected:
>>>     STUDENT
--- Typing error. "club.zbx" Line 17. Illegal type expression:
>>>     P STUDENT
--- Typing error. "club.zbx" Line 18. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 18. Set expected:
>>>     STUDENT
--- Typing error. "club.zbx" Line 18. Illegal type expression:
>>>     P STUDENT
--- Typing error. "club.zbx" Line 19. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 19. Illegal type expression:
>>>     STUDENT
--- Typing error. "club.zbx" Line 23. Type mismatch: Infix relation.
>>>     committee <= MaximumCommitteeSize
        Expected LHS type: Z
        Actual LHS type: UnknownType
        Expected RHS type: Z
        Actual RHS type: Z
--- Warning. Indefinite type in schema box.
... Type checking Schema box: JoinClub_Success. "club.zbx" Lines 27-37
--- Typing error. "club.zbx" Line 28. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 28. Set expected:
>>>     STUDENT
--- Typing error. "club.zbx" Line 28. Illegal type expression:
>>>     P STUDENT
--- Typing error. "club.zbx" Line 29. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 29. Set expected:
>>>     STUDENT
--- Typing error. "club.zbx" Line 29. Illegal type expression:
```

Full error log:

*(Full version of the log is available in **club.log**)*

```
Log opened at: Thu Dec 04 23:40:34 2014

... Initializing.
... Loading Z mathematical tools library: math0.zbx
Parsing main file: club.zbx
... Type checking Given set. "club.zbx" Line 4
... Type checking Free type definition: CLUB_NAME. "club.zbx" Line 6
... Type checking Axiom box. "club.zbx" Lines 8-11
... Type checking Free type definition: REPORT. "club.zbx" Line 13
--- Syntax error. "club.zbx" Line 22, near ";"
      Expecting: "=" "in" (infix relation symbol) "_inrel-id_"
>>>   committee subeq members ;
--- Syntax error. "club.zbx" Line 26, near "    ---------------------------
----------
"
      Expecting: "=" "in" (infix relation symbol) "_inrel-id_"
>>>         -------------------------------------

... Type checking Schema box: Club. "club.zbx" Lines 15-23
--- Typing error. "club.zbx" Line 17. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 17. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 17. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 18. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 18. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 18. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 19. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 19. Illegal type expression:
>>>   STUDENT
--- Typing error. "club.zbx" Line 23. Type mismatch: Infix relation.
>>>   committee <= MaximumCommitteeSize
      Expected LHS type: Z
      Actual LHS type: UnknownType
      Expected RHS type: Z
      Actual RHS type: Z
--- Warning. Indefinite type in schema box.
... Type checking Schema box: JoinClub_Success. "club.zbx" Lines 27-37
--- Typing error. "club.zbx" Line 28. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 28. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 28. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 29. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 29. Set expected:
>>>   STUDENT
--- Typing error. "club.zbx" Line 29. Illegal type expression:
>>>   P STUDENT
--- Typing error. "club.zbx" Line 30. Undefined name: STUDENT
--- Typing error. "club.zbx" Line 30. Illegal type expression:
>>>   STUDENT
--- Typing error. "club.zbx" Line 33. Type mismatch: Infix relation.
>>>   newmember? notin members
      Expected LHS type: [X]
```

```
      Actual LHS type: UnknownType
      Expected RHS type: (P [X])
      Actual RHS type: UnknownType
--- Typing error. "club.zbx" Line 34. Type mismatch: Infix relation.
>>>   members < MaximumClubSize
      Expected LHS type: Z
      Actual LHS type: UnknownType
      Expected RHS type: Z
      Actual RHS type: Z
--- Typing error. "club.zbx" Line 36. Type mismatch:
      LHS and RHS must have matching types.
>>>   committee' = committee
      LHS type: UnknownType
      RHS type: UnknownType
--- Warning. Indefinite type in schema box.
End of main file: club.zbx
Log written in "club.log"


Log closed at: Thu Dec 04 23:40:34 2014
```

Missed Error (simulated back)

```
Log opened at: Fri Dec 05 19:02:55 2014

... Initializing.
... Loading Z mathematical tools library: math0.zbx
Parsing main file: club.zbx
... Type checking Given set. "club.zbx" Line 4
... Type checking Free type definition: CLUB_NAME. "club.zbx" Line 6
... Type checking Axiom box. "club.zbx" Lines 8-11
... Type checking Free type definition: REPORT. "club.zbx" Line 13
... Type checking Schema box: Club. "club.zbx" Lines 15-24
... Type checking Schema box: JoinClub_Success. "club.zbx" Lines 27-37
--- Typing error. "club.zbx" Line 29. Duplicate definition of name:
committee
--- Typing error. "club.zbx" Line 36. Undefined name: committee'
End of main file: club.zbx
Log written in "club.log"


Log closed at: Fri Dec 05 19:02:55 2014
```

*(Full version of the log is available in **club.log**)*

2. Additions

*(Full version is available in **club.zbx**)*

I. InitialClub

```
      ---InitialClub-----------------------
      | Club'
      |-------
      | name' = Chess;
      | members' = { Yehan, Praminda, Grainier, Anushka, Navin, Nadil,
Sanidu };
      | committee' = { Yehan, Praminda, Navin };
      | president' = Yehan
      -------------------------------------
```

II. JoinClub total operation

```
JoinClub
========
New member can join the club successfully since:
* he/she is not a bember already,
* there are space left for hin in the club.

      ---JoinClub_Success------------------
      | Delta Club;
      | newmember? : STUDENT;
      | report! : REPORT
      |-------
      | newmember? notin members;
      | # members < MaximumClubSize;
      | members' = members || { newmember? };
      | name' = name;
      | committee' = committee;
      | president' = president;
      | report! = Success
      -------------------------------------

New member cannot join the club since:
* he/she is already a member.

      ---JoinClub_Error_Already_Member-----
      | Xi Club;
      | newmember? : STUDENT;
      | report! : REPORT
      |-------
      | newmember? in members;
      | report! = Error_Already_Member
      -------------------------------------

New member cannot join the club since:
* the club is full

      ---JoinClub_Error_No_Space-----------
      | Xi Club;
      | report! : REPORT
      |-------
      | # members = MaximumClubSize;
```

```
      | report! = Error_No_Space
      -------------------------------------
```

```
 Total JoinClub operation

      %% operation JoinClub

      JoinClub =^= JoinClub_Success \/ JoinClub_Error_Already_Member
            \/ JoinClub_Error_No_Space
```

III. LeaveClub total operation

```
LeaveClub
=========

Committee member successfully leaves the club

      ---LeaveClub_Success_Club_And_Committee---------
      | Delta Club;
      | leavingmember? : STUDENT;
      | report! : REPORT
      |-------
      | leavingmember? in committee;
      | leavingmember? /= president;
      | committee' = committee \ { leavingmember? };
      | members' = members \ { leavingmember? };
      | name' = name;
      | president' = president;
      | report! = Success
      ------------------------------------------------

Club member successfully leaves the club

      ---LeaveClub_Success_Club---------
      | Delta Club;
      | leavingmember? : STUDENT;
      | report! : REPORT
      |-------
      | leavingmember? notin committee;
      | leavingmember? in members;
      | members' = members \ { leavingmember? };
      | name' = name;
      | committee' = committee;
      | president' = president;
      | report! = Success
      ------------------------------------------------

Successful LeaveClub

      LeaveClub_Success =^= LeaveClub_Success_Club_And_Committee
            \/ LeaveClub_Success_Club

Member cannot leave the club since he/she is the president

      ---LeaveClub_Error_Is_President-------------------
      | Xi Club;
      | leavingmember? : STUDENT;
      | report! : REPORT
      |-------
      | leavingmember? = president;
```

```
      | report! = Error_Is_President
      ------------------------------------------------

Student is not in the club to leave

      ---LeaveClub_Error_Not_Member-------------------
      | Xi Club;
      | leavingmember? : STUDENT;
      | report! : REPORT
      |-------
      | leavingmember? notin members;
      | report! = Error_Not_Member
      ------------------------------------------------

Error LeaveClub

      LeaveClub_Error =^= LeaveClub_Error_Is_President
            \/ LeaveClub_Error_Not_Member

Total LeaveClub

      %% operation LeaveClub

      LeaveClub =^= LeaveClub_Success \/ LeaveClub_Error
```

IV. CommitteeMembers

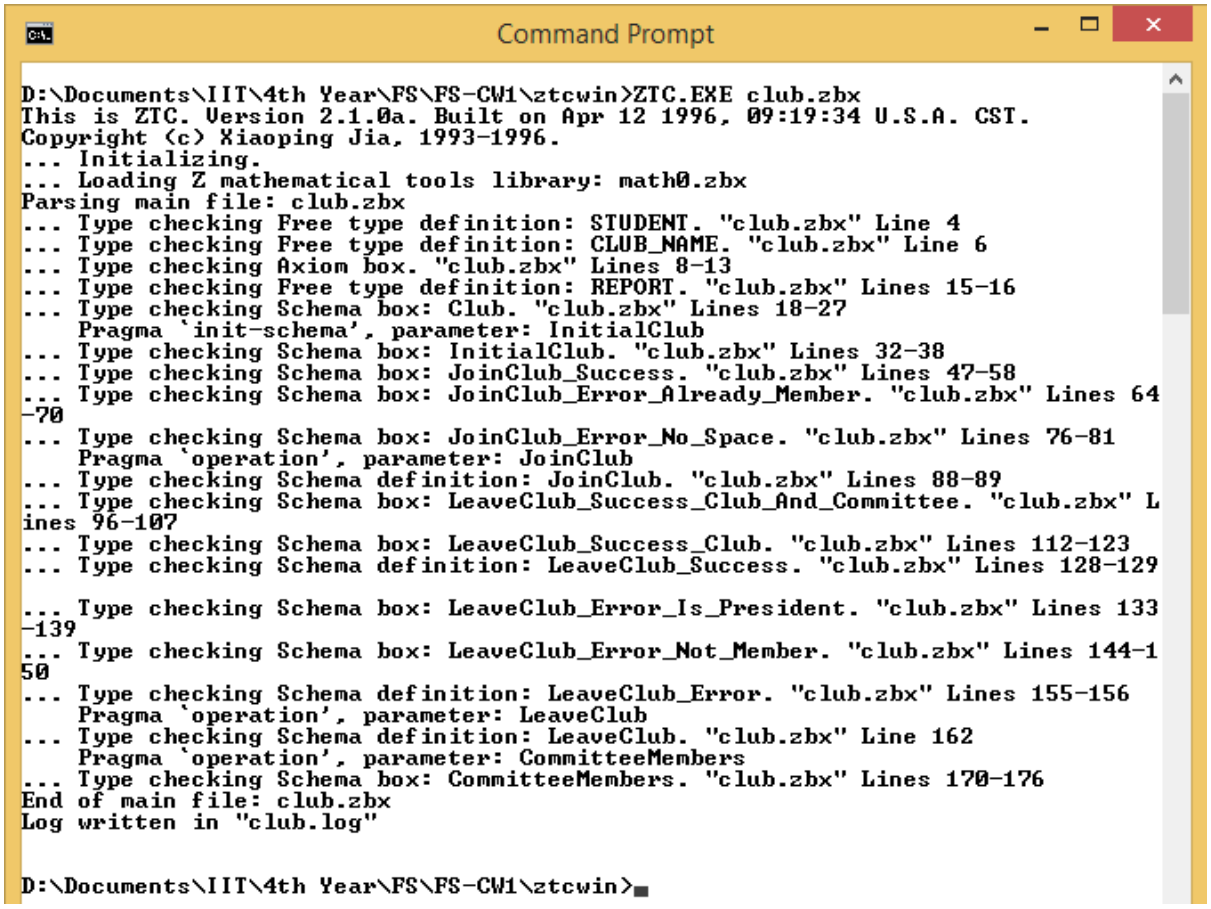
```
CommitteeMembers
================
Prints the members of the committee

      %% operation CommitteeMembers

      ---CommitteeMembers----------------------------
      | Xi Club;
      | committeemembers! : P STUDENT;
      | report! : REPORT
      |-------
      | committeemembers! = committee;
      | report! = Success
      ------------------------------------------------
```

*(Full version is available in **club.zbx**)*

3. Type checking and Animating

I. Type checking

*(Text version is available in **club.log**)*



```
D:\Documents\IIT\4th Year\FS\FS-CW1\ztcwin>ZTC.EXE club.zbx
This is ZTC. Version 2.1.0a. Built on Apr 12 1996, 09:19:34 U.S.A. CST.
Copyright (c) Xiaoping Jia, 1993-1996.
... Initializing.
... Loading Z mathematical tools library: math0.zbx
Parsing main file: club.zbx
... Type checking Free type definition: STUDENT. "club.zbx" Line 4
... Type checking Free type definition: CLUB_NAME. "club.zbx" Line 6
... Type checking Axiom box. "club.zbx" Lines 8-13
... Type checking Free type definition: REPORT. "club.zbx" Lines 15-16
... Type checking Schema box: Club. "club.zbx" Lines 18-27
    Pragma `init-schema', parameter: InitialClub
... Type checking Schema box: InitialClub. "club.zbx" Lines 32-38
... Type checking Schema box: JoinClub_Success. "club.zbx" Lines 47-58
... Type checking Schema box: JoinClub_Error_Already_Member. "club.zbx" Lines 64
-70
... Type checking Schema box: JoinClub_Error_No_Space. "club.zbx" Lines 76-81
    Pragma `operation', parameter: JoinClub
... Type checking Schema definition: JoinClub. "club.zbx" Lines 88-89
... Type checking Schema box: LeaveClub_Success_Club_And_Committee. "club.zbx" L
ines 96-107
... Type checking Schema box: LeaveClub_Success_Club. "club.zbx" Lines 112-123
... Type checking Schema definition: LeaveClub_Success. "club.zbx" Lines 128-129

... Type checking Schema box: LeaveClub_Error_Is_President. "club.zbx" Lines 133
-139
... Type checking Schema box: LeaveClub_Error_Not_Member. "club.zbx" Lines 144-1
50
... Type checking Schema definition: LeaveClub_Error. "club.zbx" Lines 155-156
    Pragma `operation', parameter: LeaveClub
... Type checking Schema definition: LeaveClub. "club.zbx" Line 162
    Pragma `operation', parameter: CommitteeMembers
... Type checking Schema box: CommitteeMembers. "club.zbx" Lines 170-176
End of main file: club.zbx
Log written in "club.log"


D:\Documents\IIT\4th Year\FS\FS-CW1\ztcwin>
```

II. Animating

*(Text version is available in **zans-club.log**)*

Loading

```
D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE
... Initializing.
... Loading Z mathematical tools library: math1.zbx

zans> load club.zbx
Parsing main file: club.zbx
... Type checking Free type definition: STUDENT. "club.zbx" Line 4
... Type checking Free type definition: CLUB_NAME. "club.zbx" Line 6
... Type checking Axiom box. "club.zbx" Lines 8-13
... Type checking Free type definition: REPORT. "club.zbx" Lines 15-16
... Type checking Schema box: Club. "club.zbx" Lines 18-27
    Pragma `init-schema', parameter: InitialClub
... Type checking Schema box: InitialClub. "club.zbx" Lines 32-38
... Type checking Schema box: JoinClub_Success. "club.zbx" Lines 47-58
... Type checking Schema box: JoinClub_Error_Already_Member. "club.zbx" Lines 64
-70
... Type checking Schema box: JoinClub_Error_No_Space. "club.zbx" Lines 76-81
    Pragma `operation', parameter: JoinClub
... Type checking Schema definition: JoinClub. "club.zbx" Lines 88-89
... Type checking Schema box: LeaveClub_Success_Club_And_Committee. "club.zbx" L
ines 96-107
... Type checking Schema box: LeaveClub_Success_Club. "club.zbx" Lines 112-123
... Type checking Schema definition: LeaveClub_Success. "club.zbx" Lines 128-129

... Type checking Schema box: LeaveClub_Error_Is_President. "club.zbx" Lines 133
-139
... Type checking Schema box: LeaveClub_Error_Not_Member. "club.zbx" Lines 144-1
50
... Type checking Schema definition: LeaveClub_Error. "club.zbx" Lines 155-156
    Pragma `operation', parameter: LeaveClub
... Type checking Schema definition: LeaveClub. "club.zbx" Line 162
    Pragma `operation', parameter: CommitteeMembers
... Type checking Schema box: CommitteeMembers. "club.zbx" Lines 170-176
End of main file: club.zbx
... Print syntax tree in debug file.
... Print syntax tree in debug file. Done!
... Unparse syntax tree in debug file.
... Unparse syntax tree in debug file. Done!
... Generate type report in debug file.

zans> list

        Club
        InitialClub
        JoinClub_Success
        JoinClub_Error_Already_Member
        JoinClub_Error_No_Space
        JoinClub
        LeaveClub_Success_Club_And_Committee
        LeaveClub_Success_Club
        LeaveClub_Success
        LeaveClub_Error_Is_President
        LeaveClub_Error_Not_Member
        LeaveClub_Error
        LeaveClub
        CommitteeMembers

zans>
```

Animating

```
D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE
zans> animate
... Initialization.
    - Search for schema classification pragmas.
      pragma `init-schema', parameter: InitialClub
      pragma `operation', parameter: JoinClub
      pragma `operation', parameter: LeaveClub
      pragma `operation', parameter: CommitteeMembers
    - Analyze state schemas.
      No `state-schema' pragma found. Attempt auto-set.
        check schema: Club
        state schema found: Club
        check schema: JoinClub_Success
        check schema: JoinClub_Error_Already_Member
        check schema: JoinClub_Error_No_Space
        check schema: JoinClub
        check schema: LeaveClub_Success_Club_And_Committee
        check schema: LeaveClub_Success_Club
        check schema: LeaveClub_Success
        check schema: LeaveClub_Error_Is_President
        check schema: LeaveClub_Error_Not_Member
        check schema: LeaveClub_Error
        check schema: LeaveClub
        check schema: CommitteeMembers
      State schema analysis done.
    - Analyze initialization schemas.
        analyze init schemaInitialClub
        init schema: InitialClub -- ok.
      Initialization schema analysis done.
    - Analyze axiomatic definitions -- ok.
    - Analyze operation schemas.
        analyze operation schema: InitialClub -- ok.
        analyze operation schema: JoinClub -- ok.
        analyze operation schema: LeaveClub -- ok.
        analyze operation schema: CommitteeMembers -- ok.
      Operation schema analysis done.
... Initializing equivalence definitions.
... Initializing global names.
### Try branch #1
    *** Statements:
        MaximumClubSize := 25;
        MaximumCommitteeSize := 10;
    *** Exit guards:
        MaximumCommitteeSize <= MaximumClubSize
        --> True
        MaximumCommitteeSize >= 0
        --> True
        MaximumClubSize >= 0
        --> True
### Branch #1 succeed.
MaximumCommitteeSize    : 10
MaximumClubSize : 25
    Initialization schema InitialClub
... Execute schema: InitialClub
### Try branch #1
    *** Statements:
        name' := Chess;
        members' := { Yehan, Praminda, Grainier, Anushka, Navin,
```

```
          Nadil, Sanidu };
        committee' := { Yehan, Praminda, Navin };
        president' := Yehan;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #1 succeed.
Schema: InitialClub
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu}
committee':     {Yehan, Praminda, Navin}
president':     Yehan

anim>
```

JoinClub: Success

D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE

```
anim> execute JoinClub
... Execute schema: JoinClub
Enter input arguments:
newmember?      -> Maura
### Try branch #1
    *** Entry guards:
        newmember? notin members
        --> True
        # members < MaximumClubSize
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        members' := members ++ { newmember? };
        name' := name;
        committee' := committee;
        president' := president;
        report! := Success;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #1 succeed.
Schema: JoinClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu}
committee:      {Yehan, Praminda, Navin}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
}
committee':     {Yehan, Praminda, Navin}
president':     Yehan
newmember?:     Maura
report!:        Success

anim> _
```

JoinClub: Error-Already a member

D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE

```
anim> execute JoinClub
... Execute schema: JoinClub
Enter input arguments:
newmember?      -> Maura
### Try branch #1
    *** Entry guards:
        newmember? notin members
        --> False
### Branch #1 fail.
### Try branch #2
    *** Entry guards:
        newmember? in members
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        report! := Error_Already_Member;
        name' := name;
        members' := members;
        committee' := committee;
        president' := president;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #2 succeed.
Schema: JoinClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
}
committee:      {Yehan, Praminda, Navin}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
}
committee':     {Yehan, Praminda, Navin}
president':     Yehan
newmember?:     Maura
report!:        Error_Already_Member

anim>
```

JoinClub: Error-No Space (Club full)

```
D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE

anim> execute JoinClub
... Execute schema: JoinClub
Enter input arguments:
newmember?      -> member26
### Try branch #1
    *** Entry guards:
        newmember? notin members
        --> True
        # members < MaximumClubSize
        --> False
### Branch #1 fail.
### Try branch #2
    *** Entry guards:
        newmember? in members
        --> False
### Branch #2 fail.
### Try branch #3
    *** Entry guards:
        # members = MaximumClubSize
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        report! := Error_No_Space;
        name' := name;
        members' := members;
        committee' := committee;
        president' := president;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #3 succeed.
Schema: JoinClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
, member9, member10, member11, member12, member13, member14, member15, member16,
 member17, member18, member19, member20, member21, member22, member23, member24,
 member25}
committee:      {Yehan, Praminda, Navin}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
, member9, member10, member11, member12, member13, member14, member15, member16,
 member17, member18, member19, member20, member21, member22, member23, member24,
```

```
 member25}
committee:      {Yehan, Praminda, Navin}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
, member9, member10, member11, member12, member13, member14, member15, member16,
 member17, member18, member19, member20, member21, member22, member23, member24,
 member25}
committee':     {Yehan, Praminda, Navin}
president':     Yehan
newmember?:     member26
report!:        Error_No_Space

anim> _
```

LeaveClub: Success-Committee member leaving

```
anim> execute LeaveClub
... Execute schema: LeaveClub
Enter input arguments:
leavingmember?  -> Navin
### Try branch #1
    *** Entry guards:
        leavingmember? in committee
        --> True
        leavingmember? /= president
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        committee' := committee \ { leavingmember? };
        members' := members \ { leavingmember? };
        name' := name;
        president' := president;
        report! := Success;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #1 succeed.
Schema: LeaveClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Navin, Nadil, Sanidu, Maura
, member9, member10, member11, member12, member13, member14, member15, member16,
 member17, member18, member19, member20, member21, member22, member23, member24,
 member25}
committee:      {Yehan, Praminda, Navin}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, Maura, membe
r9, member10, member11, member12, member13, member14, member15, member16, member
17, member18, member19, member20, member21, member22, member23, member24, member
25}
committee':     {Yehan, Praminda}
president':     Yehan
leavingmember?:         Navin
report!:        Success

anim>
```

LeaveClub: Success-Club member leaving

```
D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE

anim> execute LeaveClub
... Execute schema: LeaveClub
Enter input arguments:
leavingmember?  -> Maura
### Try branch #1
    *** Entry guards:
        leavingmember? in committee
        --> False
### Branch #1 fail.
### Try branch #2
    *** Entry guards:
        leavingmember? notin committee
        --> True
        leavingmember? in members
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        members' := members \ { leavingmember? };
        name' := name;
        committee' := committee;
        president' := president;
        report! := Success;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #2 succeed.
Schema: LeaveClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, Maura, membe
r9, member10, member11, member12, member13, member14, member15, member16, member
17, member18, member19, member20, member21, member22, member23, member24, member
25}
committee:      {Yehan, Praminda}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, member9, mem
ber10, member11, member12, member13, member14, member15, member16, member17, mem
ber18, member19, member20, member21, member22, member23, member24, member25}
committee':     {Yehan, Praminda}
president':     Yehan
leavingmember?:         Maura
report!:        Success

anim>
```

LeaveClub: Error-President tries to leave

```
D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE

anim> execute LeaveClub
... Execute schema: LeaveClub
Enter input arguments:
leavingmember?  -> Yehan
### Try branch #1
    *** Entry guards:
        leavingmember? in committee
        --> True
        leavingmember? /= president
        --> False
### Branch #1 fail.
### Try branch #2
    *** Entry guards:
        leavingmember? notin committee
        --> False
### Branch #2 fail.
### Try branch #3
    *** Entry guards:
        leavingmember? = president
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        report! := Error_Is_President;
        name' := name;
        members' := members;
        committee' := committee;
        president' := president;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #3 succeed.
Schema: LeaveClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, member9, mem
ber10, member11, member12, member13, member14, member15, member16, member17, mem
ber18, member19, member20, member21, member22, member23, member24, member25}
committee:      {Yehan, Praminda}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, member9, mem
ber10, member11, member12, member13, member14, member15, member16, member17, mem
ber18, member19, member20, member21, member22, member23, member24, member25}
committee':     {Yehan, Praminda}
```

```
president':     Yehan
leavingmember?:         Yehan
report!:        Error_Is_President

anim>
```

LeaveClub: Error-Not a member to leave

```
D:\Documents\IIT\4th Year\FS\FS-CW1\zanswin\ZANS.EXE

anim> execute LeaveClub
... Execute schema: LeaveClub
Enter input arguments:
leavingmember?  -> someone
### Try branch #1
    *** Entry guards:
        leavingmember? in committee
        --> False
### Branch #1 fail.
### Try branch #2
    *** Entry guards:
        leavingmember? notin committee
        --> True
        leavingmember? in members
        --> False
### Branch #2 fail.
### Try branch #3
    *** Entry guards:
        leavingmember? = president
        --> False
### Branch #3 fail.
### Try branch #4
    *** Entry guards:
        leavingmember? notin members
        --> True
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        report! := Error_Not_Member;
        name' := name;
        members' := members;
        committee' := committee;
        president' := president;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #4 succeed.
Schema: LeaveClub
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, member9, mem
ber10, member11, member12, member13, member14, member15, member16, member17, mem
ber18, member19, member20, member21, member22, member23, member24, member25}
committee:      {Yehan, Praminda}
president:      Yehan
```

```
leavingmember?:         someone
report!:        Error_Not_Member

anim>
```

CommitteeMembers

```
anim> execute CommitteeMembers
... Execute schema: CommitteeMembers
### Try branch #1
    *** Entry guards:
        # members <= MaximumClubSize
        --> True
        committee subseteq members
        --> True
        # committee <= MaximumCommitteeSize
        --> True
        president in committee
        --> True
    *** Statements:
        committeemembers! := committee;
        report! := Success;
        name' := name;
        members' := members;
        committee' := committee;
        president' := president;
    *** Exit guards:
        # members' <= MaximumClubSize
        --> True
        committee' subseteq members'
        --> True
        # committee' <= MaximumCommitteeSize
        --> True
        president' in committee'
        --> True
### Branch #1 succeed.
Schema: CommitteeMembers
name:   Chess
members:        {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, member9, mem
ber10, member11, member12, member13, member14, member15, member16, member17, mem
ber18, member19, member20, member21, member22, member23, member24, member25}
committee:      {Yehan, Praminda}
president:      Yehan
name':  Chess
members':       {Yehan, Praminda, Grainier, Anushka, Nadil, Sanidu, member9, mem
ber10, member11, member12, member13, member14, member15, member16, member17, mem
ber18, member19, member20, member21, member22, member23, member24, member25}
committee':     {Yehan, Praminda}
president':     Yehan
committeemembers!:      {Yehan, Praminda}
report!:        Success

anim> _
```

*(Text version is available in **zans-club.log**)*