Name: Darwish bin Mat Zain ID: CB19102

Defining the data type used in the Z-Specification

─ [STUDENT,ADVISOR,COURSE,SECTION,BOOL] └

Defining the value for data type BOOL

```
|
 True,False : BOOL
└
```

Defining system schema

```
┌ SystemSchema
        Student: ℙ STUDENT
        Studentcourse: STUDENT ℙ □ COURSE
        Studentsection: STUDENT ⇸ ℙ COURSE ⇸ ℙ SECTION
        Courseapproved: STUDENT ⇸ ℙ COURSE ⇸ BOOL
        Studentadvisor: STUDENT ⇸ ADVISOR
        Studentcredit: STUDENT ⇸ ℕ
        Advisor:  ℙ ADVISOR
        Course: ℙ COURSE
        Section: COURSE ⇸ ℙ SECTION
        Credit: COURSE ⇸ ℕ
        Maxcredit: ℕ
         Studentlogged : ℙ STUDENT
        Advisorlogged : ℙ ADVISOR
|
        Maxcredit = 20
└
```

System schema defines **Student** as student identification (student ID), **Studentcourse** as the courses registered to a student, **Studentsection** as the section of the courses regisered to a student, **Courseapproved** as boolean value of the courses registered to a student, **Courseadvisor** as advisor identification (staff ID) that is assigned to a student, **Studentcredit** as the total credit hour for a student, **Advisor** as the adivisor identification (staff ID), **Course** as courses' code, **Section** as the section of a course, **Credit** as the credit hour for a course, **Maxcredit** as the maximum credit hours allowed for the semester, **Studentlogged** as the student identification (student ID) for students logged in into the system and **Advisorlogged** as the advisor identification (staff ID) for advisors logged in into the system.

The value for **Maxcredit** is set to 20.

Defining initialization schema

```
┌ InitSystem
        ΔSystemSchema
|
        Student = ∅
```

Studentcourse = ∅
Studentsection = ∅
Courseapproved = ∅
Studentadvisor = ∅
Studentcredit = ∅
Studentcourse = ∅
Advisor = ∅
Course = ∅
Section = ∅
Credit = ∅
Studentlogged = ∅
Advisorlogged = ∅

When the system is initiated, the data set for **Student**, **Studentcourse**, **Studentsection**, **Courseapproved**, **Studentadvisor**, **Studentcredit**, **Studentcourse**, **Advisor**, **Course**, **Section**, **Credit**, **Studentlogged** and **Advisorlogged** is set to empty set (∅)

Defining schema for login (Student and **Advisor)**

Studentlogin
ΔSystemSchema
student?: STUDENT
Studentlogged′: ℙ STUDENT

student? ∈ Student
Studentlogged = ∅
Studentlogged′ = Studentlogged ∪ {student?}

Advisorlogin
ΔSystemSchema
advisor?: ADVISOR
Advisorlogged′: ℙ ADVISOR

advisor? ∈ Advisor
Advisorlogged = ∅
Advisorlogged′ = Advisorlogged ∪ {advisor?}

The system request input for **student?** (student ID for student) or **advisor?** (staff ID for advisor). The system check if **student?** is an element of **Student** or **advisor?** is an element of **Advisor**, then check if **Studentlogged** or **Advisorlogged** is an empty set and the **Studentlogged** or **Advisorlogged** will be set to **student?** or **advisor?**.

Defining schema for logout (Student and **Advisor)**

Studentlogout
ΔSystemSchema
student?: STUDENT

```
|
        student? ∈ Student
                Studentlogged ≠ ∅
                        Studentlogged = ∅
└
┌ Advisorlogout
 ΔSystemSchema
 advisor?: ADVISOR
|
        advisor? ∈ Advisor
                Advisorlogged ≠ ∅
                        Advisorlogged = ∅
└
```

The system request input for **student?** (student ID for student) or **advisor?** (staff ID for advisor). The system check if **student?** is an element of **Student** or **advisor?** is an element of **Advisor**, then check if **Studentlogged** or **Advisorlogged** is not an empty set. Then, **Studentlogged** or **Advisorlogged** will be set to an empty set.

Defining schema for course registration

```
┌ CourseRegister
 ΔSystemSchema
 ΞStudentlogin
 studentlogged?: STUDENT
 course? : COURSE
 section?: ℙ SECTION
|
        studentlogged? ∈ Studentlogged
                course? ∈ Course
                        section? ∈ ℙ Section
                                Credit(course?) + Studentcredit( studentlogged?) ≤ Maxcredit
                                        Studentcourse′ = Studentcourse ⊕ {studentlogged? ↦
{ course? }}
                                        Studentsection′ = Studentsection ⊕ { studentlogged? ↦
{ course? ↦ section? } }
                                        Studentcredit′(studentlogged?) =
Studentcredit(studentlogged?) + Credit(course?)
└
```

The system request inputs for **studentlogged?**, **course?**, **section?.** The system check if the student logged in into the system, then check if **course?** exists in course data set and **section?** exist in section data set, then check if the course credit is added to student's total credit hour didn't exceed 20. Then the new course is added to **Studentcourse**, the section is added to **Studentsection** and course credit hour is added to **Studentcredit**.

Defining schema for course withdrawal

```
┌ Coursewithdraw
  ΔsystemSchema
```

```
ΞStudentlogin
studentlogged?: STUDENT
course? : COURSE
section?: ℙ SECTION
|
studentlogged? ∈ Studentlogged
        course? ∈ Course
                section? ∈ ℙ Section
                        Studentcourse′ = Studentcourse ⊖ {studentlogged? ↦
{ course? }}
                        Studentsection′ = Studentsection ⊖ { studentlogged? ↦
{ course ↦ section? } }
                        Studentcredit′(studentlogged?) = Studentcredit(studentlogged?)
− Credit(course?)
└
```

The system request inputs for **studentlogged?**, **course?**, **section?.** The system check if the student logged in into the system, then check if **course?** exists in course data set and **section?** exist in section data set. Then the system remove the course from **Studentcourse**, the section is removed from **Studentsection** and course credit hour is subtracted from **Studentcredit**.

Defining schema for course editting

```
┌ Courseedit
ΔSystemSchema
ΞStudentlogin
studentlogged?: STUDENT
course? : COURSE
section?: ℙ SECTION
|
studentlogged? ∈ Studentlogged
      course? ∈ Course
              section? ∈ ℙ Section
                      Studentsection′ = Studentsection ⊖ { studentlogged? ↦ { course? ↦
section? } }
                      Studentsection′ = Studentsection ⊕ { studentlogged? ↦ { course? ↦
section? } }

└
```

The system request inputs for **studentlogged?**, **course?**, **section?.** The system check if the student logged in into the system, then check if **course?** exists in course data set and **section?** exist in section data set. Then the section for the course is removed from **Studentsection** and the new section is added to **Studentsection**.

Defining schema for advisor notification

```
┌ Advisornotify
ΔSystemSchema
```

```
ΞAdvisorlogin
advisorlogged?: ADVISOR
studentcheck? : STUDENT
studentinfo!: STUDENT
|
        advisorlogged? ∈ Advisorlogged
                advisorlogged? = Studentadvisor(studentcheck?)
                        Courseapproved(studentcheck?)(Course) = False

└
```

The system request inputs for **advisorlogged?** and **studentcheck?**. The system checks if the advisor logged in into the system, then check if the advisor is the advisor of the **studentcheck?**, then check if the boolean value of **Courseapproved** for **studentcheck** is false.

Defining schema for advisor approval

```
┌Advisorapprove
 ΔSystemSchema
advisorlogged?: ADVISOR
studentcheck? : STUDENT
coursecheck?: COURSE
|
        advisorlogged? ∈ Advisorlogged
                advisorlogged? = Studentadvisor(studentcheck?)
                        coursecheck? ∈ Course
                                Courseapproved′ = Courseapproved ⊕ {studentcheck? ↦
{{coursecheck?} ↦ True}}
└
```

The system requests inputs for **advisorlogged?**, **studentcheck?** and **coursecheck.** The system checks if the advisor is logged in, then checks if the advisor is the advisor of the **studentcheck?**, then set the boolean value of **Courseapproved** for **studentcheck?** to true.

Defining schema for student view

```
┌ Studentdisplay
        ΞSystemSchema
        ΞStudentlogin
        studentlogged?: STUDENT
        studentcourse! : STUDENT ⇸ ℙ COURSE
        studentsection! : STUDENT ⇸  ℙ COURSE ⇸ ℙ SECTION
        courseapproved! : STUDENT ⇸ ℙ COURSE ⇸ BOOL
        studentadvisor! : STUDENT ⇸ ADVISOR
        studentcredit! : STUDENT ⇸ ℕ
|
        studentlogged? ∈ Studentlogged
                studentcourse!(studentlogged?) = Studentcourse(studentlogged?)
                studentsection!(studentlogged?) = Studentsection(studentlogged?)
```

courseapproved!(studentlogged?) = Courseapproved(studentlogged?)
studentadvisor!(studentlogged?) = Studentadvisor(studentlogged?)
studentcredit!(studentlogged?) = Studentcredit(studentlogged?)

└

The system request input for **studentlogged?**. Then the system check if the student are logged in. Then the system outputs **Studentcourse**, **Studentsection**, **Courseapproved**, **Studentadvisor** and **Studentcredit** of the **studentlogged?.**

Defining schema for advisor view

┌ Advisordisplay
ΞSystemSchema
ΞAdvisorlogin
advisorlogged?:ADVISOR
studentcheck? : STUDENT
studentcourse! : STUDENT ⇸ ℙ COURSE
studentsection! : STUDENT ⇸ ℙ COURSE ⇸ ℙ SECTION
courseapproved! : STUDENT ⇸ ℙ COURSE ⇸ BOOL
studentcredit! : STUDENT ⇸ ℕ

|

advisorlogged? ∈ Advisorlogged
advisorlogged? = Studentadvisor(studentcheck?)
studentcourse!(studentcheck?) = Studentcourse(studentcheck?)
studentsection!(studentcheck?) = Studentsection(studentcheck?)
courseapproved!(studentcheck?) = Courseapproved(studentcheck?)
studentcredit!(studentcheck?) = Studentcredit(studentcheck?)

└

The system request input for **advisorlogged?** and **studentcheck?.** The system check if the advisor is logged in, then check if the advisor is the advisor for **studentcheck?**. Then the system outputs the **Studentcourse**, **Studentsection**, **Courseapproved** and **Studentcredit** of **studentcheck?**.