Task 1

Test Case 1: test_tokenGenerator

Description: Ensures the functionality of login. When user sends the log in request ,ShibbolethAuth.tokenGenerator() is invoked, and generates the token with expected type (STUDENT, ADMIN, BOTH, UNDEFINED).

Test Inputs:

1. x500=’Abc1001’,password=’1000’(Assume this x500 exists in the X500ACCOUNT colume of SHIBBOLETHAUTH Table, with the entry’s Type=’STUDENT’ and this password equals the entry’s X500ACCOUNT value)
2. x500=’Abc1005’,password=’5000’(Assume this x500 exists in the X500ACCOUNT colume of SHIBBOLETHAUTH Table, with the entry’s Type=’ADMIN’ and this password equals the entry’s X500ACCOUNT value)
3. x500=’Abc1002’,password=’2000’(Assume this x500 exists in the X500ACCOUNT colume of SHIBBOLETHAUTH Table, with the entry’s Type=’BOTH’ and this password equals the entry’s X500ACCOUNT value)
4.x500=’Abc1111’,password=’12345’(Assume this x500 doesn’t exist in the X500ACCOUNT colume of SHIBBOLETHAUTH Table)
5.x500=’Abc1001’,password=’2000’(Assume this x500 exists in the X500ACCOUNT colume of SHIBBOLETHAUTH Table but this password doesn’t equal the entry’s X500ACCOUNT value)

Expected Results:

1.Token object with type=STUDENT
2.Token object with type=ADMIN
3.Token object with type=BOTH
4.Token object with type=UNDEFINED
5.Token object with type=UNDEFINED

Dependencies: None

Initialization: All x500 and passwords are stored in the SUBBOLETHAUTH Table.

Test Step:

1.The uses send the login request by invoke ShibbolethAuth.tokenGenerator() method with x500 and password

2.The correct Token object is returned

Test Case 2: test_queryclass

Description: Ensures the functionality of queryclass. When user sends the queryclass request ,ISCRS.queryclass() is invoked, it takes *courseID, courseName,location,term, department, classType, instructorName* as input and return a list of Arraylist of expected result. Every Arraylist stores the following information *ID,NAME,CREDITS,CAPACITY,TERM,FIRSTDAY,LASTDAY,CLASSBEGINTIME,CLASSENDTIME,ROUTINES,LOCATION,TYPE,PREREQUISITE,DESCRIPTION,DEPARTMENNT* in order. If query is not valid or no matching results, the method returns an empty list.

Test Inputs:

1. -1,'Advanced Algorithm','East Campus','Fall 2015','','','(all required criteria included with none optional criteria)

2.-1,'Advanced Algorithm','East Campus','Fall 2015','CS','','(all required criteria included with one optional criteria department)

3.-1,'Software Engineering','East Campus','Fall 2015','','','Kevin Wendt'(all required criteria included with one optional criteria instructorName)

4.2,'Software Engineering','East Campus','Fall 2015','','','Kevin Wendt'(all required criteria included with two optional criteria courseID and instructorName)

5.-1,'','','','','','','(None criteria included)

6.-1,'','','Fall 2015','','','(not all required criteria included)

Expected Results:

1.For 1-4, a list of Arraylist as specified in description or empty list if no matching results

2.For 5-6, empty list

Dependencies: None

Initialization: Course Table, Instructor TABLE and InstructorAndCourse Table stored in the database

Test Step:

1. The uses send the queryclass request by invokeISCRS.queryclass() method with inputs as in test inputs
2. Expected results returned

Test Case 3: test_queryInstructor

Description: Ensures the functionality of queryInstructor. When user sends the queryInstructor request ,ISCRS.queryInstructor() is invoked. It takes *token, instructorID* as input and return a list of Arraylist of expected result. The list should only contain one arraylist as instructorID is unique, and the arraylist stores the following information *ID,FIRSTNAME,LASTNAME,DATEOFBIRTH,,GENDER,TITLE,SALARY,DEPARTMENT* in order. If query is not valid or no matching results, the method returns an empty list.

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),1* (Assume the instructor with ID=1 exists in database)
2. *token(id=1,type=BOTH,timestamp="2015.08.01.17.30.05"),1* (Assume the instructor with ID=1 exists in database)
3. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),10* (Assume the instructor with ID=10 doesn't exist in database)
4. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),1* (the token type is not ADMIN or BOTH)
5. *token(type=STUDENT,timestamp="2015.08.01.17.30.05"),1* (the token type is not ADMIN or BOTH)

Expected Results:

1.For 1-2, return a list of Arraylist as specified in description
2.For 3-5, return a empty list

Dependencies: 1

Initialization: Instructor Table is stored in the database

Test Step:

1.The user sends log in request and a token object generated

2.The user sends the queryInstructor request by invokeISCRS.queryInstructor() method with inputs as specified in test inputs

3.Expected list returned

Test Case 4: test_studentAddClass

Description: Ensures the functionality of student Add Class. When user sends the studentAddClass request ,ISCRS.studentAddClass() is invoked. It takes *token, courseID, grading, courseTerm* as input and return a boolean value(true is student successfully adds the class, false otherwise).

Test Inputs:

1. *token(id=1,type=STUDENT,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (Assume the course with ID=3 is not full and the student with USERID=1 will not exceed 30 credits after the adding the course)
2. *token(id=2,type=BOTH,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (Assume the course with ID=3 is not full and the student with USERID=2 will not exceed 30 credits after the adding the course)
3. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (the token type is not STUDENT or BOTH)
4. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (the token type is not STUDENT or BOTH)
5. *token(id=1,type=STUDENT,timestamp="2015.08.01.17.30.05"),4,"A-F","Fall 2015"* (Assume the course with ID=4 is not full and the student with USERID=1 will exceed 30 credits after the adding the course)
6. *token(id=1,type=STUDENT,timestamp="2015.08.01.17.30.05"),5,"A-F","Fall 2015"*(Assume the course with ID=5 is full)
7. *token(id=1,type=STUDENT,timestamp="2015.09.05.17.30.05"),3,'A-F','Fall 2015'* (Timeframe has passed)

Expected Results:

1.For 1-2, return true
2.For 3-7, return false

Dependencies: 1

Initialization: Course Table, Student TABLE and StudentAndCourse Table are stored in the database

Test Step:

1.The user sends log in request and a token object generated
2.The user sends the student add class request by invokeISCRS.studentAddClass() method with inputs as specified in test inputs
3.Expected boolean value returned

Test Case 5: test_studentEditClass

Description: Ensures the functionality of student Edit Class. When user sends the studentEditClass request ,ISCRS.studentEditClass() is invoked. It takes *token, courseID, grading, courseTerm* as input and return a boolean value(true is student successfully edits the class, false otherwise).

Test Inputs:

1. *token(id=1,type=STUDENT,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (all required criteria included)
2. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (the token type is not STUDENT or BOTH)
3. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3,"A-F","Fall 2015"* (the token type is not STUDENT or BOTH)
4. *token(id=1,type=STUDENT,timestamp="2015.09.05.17.30.05"),3,'A-F','Fall 2015'* (Timeframe has passed)

Expected Results:

1.For 1, return true

2.For 2-4, return false

Dependencies: 1

Initialization: Course Table, Student Table and StudentAndCourse Table are stored in the database

Test Step:

1.The user sends log in request and a token object generated

2.The user sends the student edit class request by invokeISCRS.studentEditClass() method with inputs as specified in test inputs

3.Expected boolean value returned

Test Case 6: test_studentDropClass

Description: Ensures the functionality of student Drop Class. When user sends the studentDropClass request ,ISCRS.studentDropClass() is invoked. It takes *token, courseID* as input and return a boolean value(true is student successfully drops the class, false otherwise).

Test Inputs:

1. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),3* (all required criteria included)
2. token*(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),4* (Assume course has is not in the sutdent's registered course list)
3. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05"),3* (the token type is not STUDENT or BOTH)
4. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3* (the token type is not STUDENT or BOTH)
5. *token(id=1,type=STUDENT,timestamp="2015.09.05.17.30.05"),3* (Timeframe has passed)

Expected Results:

1.For 1, return true
2.For 2-5, return false

Dependencies: 1

Initialization: Course Table Student Table and StudentAndCourse Table are stored in the database

Test Step:

1.The user sends log in request and a token object generated

2.The user sends the student drop class request by invokeISCRS.studentDropClass() method with inputs as specified in test inputs

3.Expected boolean value returned

Test Case 7: test_queryStudentPersonalData

Description: Ensures the functionality of queryStudentPersonalData. When user sends the queryStudentPersonalData request ,ISCRS.queryStudentPersonalData() is invoked, it takes *token, studentID* as input and return a list of Arraylist of expected result. Every Arraylist stores the following information *ID,FIRSTNAME, LASTNAME, DATEOFBIRTH, TYPE, GENDER, ADVISOR, CREDITS, DEPARTMENT* in order. If query is not valid or no matching results, the method returns an empty list.

Test Inputs:

1. *token(id=0,type=STUDENT, timestamp="2015.08.01.17.30.05"), 0* (student query student)
2. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05"), 0* (admin query student)
3. *token(id=0,type=STUDENT, timestamp="2015.08.01.17.30.05"), 1* (a student query another student's personal data)
4. *token(id=0,type=UNDEFINED,timestamp="2015.08.01.17.30.05")*, 0 (the token type is not STUDENT, ADMIN or BOTH)
5. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05")*, 10 (no such student)

Expected Results:

1.For 1-2, a list of Arraylist as specified in description or empty list if no matching results
2.For 3-5, empty list

Dependencies: None

Initialization: Student Table and Administrator Table stored in the database

Test Step:

1.The user sends log in request and a token object generated
2.The user sends the queryStudentPersonalData request by invokeISCRS.queryStudentPersonalData() method with inputs as specified in test inputs
3.Expected results returned

Test Case 8: test_queryStudentRegistrationHistory

Description: Ensures the functionality of queryStudentRegistrationHistory. When user sends the queryStudentRegistrationHistory request ,ISCRS.queryStudentRegistrationHistory() is invoked, it takes *token, studentID* as input and return a list of Arraylist of expected result. Every Arraylist stores the following information *CLASSID, CLASSNAME, REGISTRATION TIME, CREDITS* in order. If query is not valid or no matching results, the method returns an empty list.

Test Inputs:

1. *token(id=0,type=STUDENT, timestamp="2015.08.01.17.30.05"), 0* (student query student)
2. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05"), 0* (admin query student)
3. *token(id=0,type=STUDENT, timestamp="2015.08.01.17.30.05"), 1* (a student query another student)
4. *token(id=0,type=UNDEFINED,timestamp="2015.08.01.17.30.05")*, 0 (the token type is not STUDENT, ADMIN or BOTH)
5. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05")*, 10 (no such student)

Expected Results:

1.For 1-2, a list of Arraylist as specified in description or empty list if no matching results
2.For 3-5, empty list

Dependencies: None

Initialization: Student Table, StudentAndCourse Table, Course Table and Administrator Table stored in the database

Test Step:

1.The user sends log in request and a token object generated
2.The user sends the queryStudentRegistrationHistoryrequest by invokeISCRS.queryStudentRegistrationHistory() method with inputs as specified in test inputs
3.Expected results returned

Test Case 9: test_adminAddClass

Description: Ensures the functionality of admin Add Class. When user sends the adminAddClass request ,ISCRS.adminAddClass() is invoked. It takes *token, courseID, courseName, courseCredits, courseCapacity, term, instructorID, firstDay, lastDay, classBeginTime, classEndTime, weekDays, location, type, prerequisite, description, department* as input and return a boolean value(true is admin successfully adds the class, false otherwise).

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"* (Assume the course with ID=3 is empty)
2. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "" and "CS"* (Assume the course with ID=3 is empty and description is empty)
3. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),1, "Operating Systems", 4, 2, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"* (Assume the course with ID=1 is not empty)
4. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"* (the token type is not ADMIN or BOTH)
5. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"* (the token type is not ADMIN or BOTH)
6. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 5, 2, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"* (credits larger than 4)
7. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 31, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"* (capacity larger than 30)
8. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 31, "Fall2015", 1, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "a", "Intro to OS" and "CS"* (type is not lecture or seminar)
9. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "", 4, 29, "", 1, "", "", "", "", "", "", "", "", ""* (required field is empty)

Expected Results:

1.For 1-2, return true

2.For 3-9, return false

Dependencies: 1

Initialization: Administrator Table, Course Table and StudentAndCourse Table are stored in the database

Test Step:

1.The user sends log in request and a token object generated

2.The user sends the admin add class request by invokeISCRS.admintAddClass() method with inputs as specified in test inputs

3.Expected boolean value returned

Test Case 10: test_adminDropStudentRegisteredClass

Description: Ensures the functionality of admin Drop Student Registered Class. When user sends the adminDropStudentRegisteredClass request ,ISCRS.adminDropStudentRegisteredClass() is invoked. It takes *token, studentID, courseID* as input and return a boolean value(true is admin successfully drops the student from the class, false otherwise).

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),1,1(Assume the student with ID=1 has course with ID=1 in his registered course list)*
2. *token(id=1,type=BOTH,timestamp="2015.08.01.17.30.05"),1,1((Assume the student with ID=1 has course with ID=1 in his registered course list)*
3. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),1,10((Assume the student with ID=1 has no course with ID=10 in his registered course list)*
4. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),3,1*

(the token type is not ADMIN or BOTH)

5. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3,1*

(the token type is not ADMIN or BOTH)

Expected Results:

1.For 1-2, return true

2.For 3-5, return false

Dependencies: 1

Initialization: Course Table, Administrator Table, StudentAndCourse Table are stored in the database

Test Step:

1.The user sends login request and a token object generated

2.The user sends the admin drop student registered class request by invokeISCRS.adminDropStudentRegisteredClass() method with inputs as specified in test inputs

3.Expected boolean value returned

Test Case 11: test_adminEditStudentRegisteredClass

Description: Ensures the functionality of admin Edit Student Registered Class. When user sends the adminEditStudentRegisteredClass request ,ISCRS.adminEditStudentRegisteredClass() is invoked. It takes *token, studentID, courseID, grading, courseTerm* as input and return a boolean value(true is admin successfully edits the information of student registered class, false otherwise).

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),1,1,"S/N", "Fall2015"*
(Assume the student with ID=1 has course with ID=1 in his registered course list)
2. *token(id=2,type=BOTH,timestamp="2015.08.01.17.30.05"),1,1,"S/N", "Fall2015"*
(Assume the student with ID=1 has course with ID=1 in his registered course list)
3. *token(id=2,type=ADMIN,timestamp="2015.08.01.17.30.05"),1,10,"S/N", "Fall2015"*
(Assume the student with ID=1 has no course with ID=10 in his registered course list)
4. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),3,1,"A-F","Fall 2015"*
(the token type is not ADMIN or BOTH)
5. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3,1,"A-F","Fall 2015"*
(the token type is not ADMIN or BOTH)

Expected Results:

1.For 1-2, return true
2.For 3-5, return false

Dependencies: 1

Initialization: Course Table, Administrator Table, StudentAndCourse Table are stored in the database

Test Step:

1.The user sends login request and a token object generated
2.The user sends the admin edit student registered class request by invokeISCRS.adminEditStudentRegisteredClass() method with inputs as specified in test inputs
3.Expected boolean value returned

Test Case 12: test_adminAddStudentToClass

Description: Ensures the functionality of admin Add Student to Class. When user sends the adminAddStudentToClass request ,ISCRS.adminAddStudentToClass() is invoked. It takes *token, studentID, courseID, grading, courseTerm* as input and return a boolean value(true is admin successfully adds the student to the class, false otherwise).

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),1,1,"A-F", "Fall2015"*
(Assume the course with ID=1 is not full and the student with USERID=1 will not exceed 30 credits after the adding the course)
2. *token(id=2,type=BOTH,timestamp="2015.08.01.17.30.05"),1,1,"A-F", "Fall2015"*
(Assume the course with ID=1 is not full and the student with USERID=1 will not exceed 30 credits after the adding the course)
3. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),3,1,"A-F","Fall 2015"*
(the token type is not ADMIN or BOTH)
4. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3,1,"A-F","Fall 2015"*
(the token type is not ADMIN or BOTH)
5. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),4,1,"A-F","Fall 2015"*
(Assume the course with ID=4 is not full and the student with USERID=1 will exceed 30 credits after the adding the course)
6. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),5,1,"A-F","Fall 2015"*
(Assume the course with ID=5 is full)

Expected Results:

1.For 1-2, return true
2.For 3-6, return false

Dependencies: 1

Initialization: Course Table, Administrator Table, StudentAndCourse Table are stored in the database

Test Step:

1.The user sends log in request and a token object generated
2.The user sends the admin add student to class request by invokeISCRS.adminAddStudentToClass() method with inputs as specified in test inputs
3.Expected boolean value returned

Test Case 13: test_adminEditClass

Description: Ensures the functionality of administrator Edit Class. When user sends the admimEditClass request ,ISCRS.adminEditClass() is invoked. It takes *token, courseID, courseName, courseCredits, instructorID, firstDay, lastDay, classBeginTime, classEndTime, weekDays, location, type, prerequisite, description and department* as input and return a boolean value(true is admin successfully edits the class, false otherwise).

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"*

(Assume the course with ID=3 is empty)

2. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),-1, "", -1, -1, "", "", "", "", "", "", "", "", "", "good class" and ""*

(Assume the course with ID=3 is registered by some students)

3. *token(id=2,type=BOTH,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"*

(Assume the course with ID=3 is empty)

4. *token(id=1,type=BOTH,timestamp="2015.08.01.17.30.05"), -1, "", -1, -1, "", "", "", "", "", "", "", "", "", "good class" and ""*

(Assume the course with ID=3 is registered by some students)

5. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"*

(the token type is not ADMIN or BOTH)

5. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3, "Operating Systems", 4, 2, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"*

(the token type is not ADMIN or BOTH)

6. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),4, "Operating Systems", 4, 2, "20150913", "20151208", "09:30", "10:45", "Tu, Th", "ME 321", "Lecture", "Intro to OS" and "CS"*

(Assume the course with ID=4 is registered by some students)

7. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),5, -1, "", -1, -1, "", "", "", "", "", "", "", "", "", "" and ""*

(Assume the course with ID=5 is registered by some students)

Expected Results:

1.For 1-4, return true
2.For 5-7, return false

Dependencies: 1

Initialization: Administrator Table, Course Table, Student Table, Instructor Table, InstructorAndCourse Table and StudentAndCourse Table are stored in the database

Test Step:

1.The user sends login request and a token object generated

2.The user sends the admin edit class request by invoke ISCRS.adminEditClass() method with inputs as specified in test inputs

3.Expected boolean value returned

Test Case 14: test_adminDeleteClass

Description: Ensures the functionality of administrator Delete Class. When user sends the admimDeleteClass request ,ISCRS.adminDeleteClass() is invoked. It takes *token* and *courseID* as input and return a boolean value(true is admin successfully deletes the class, false otherwise). The course can be successfully deleted if no student has registered the class.

Test Inputs:

1. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),3*
(Assume the course with ID=3 is empty)
2. *token(id=2,type=BOTH,timestamp="2015.08.01.17.30.05"),3*
(Assume the course with ID=3 is empty)
3. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05"),3*
(the token type is not ADMIN or BOTH)
4. *token(type=UNDEFINED,timestamp="2015.08.01.17.30.05"),3*
(the token type is not ADMIN or BOTH)
5. *token(id=1,type=ADMIN,timestamp="2015.08.01.17.30.05"),4*
(Assume the course with ID=4 is registered by some students)

Expected Results:

1.For 1-2, return true
2.For 3-5, return false

Dependencies: 1

Initialization: Administrator Table, Course Table, Student Table and StudentAndCourse Table are stored in the database

Test Step:

1.The user sends login request and a token object generated

2.The user sends the admin delete class request by invoke ISCRS.adminDeleteClass() method with inputs as specified in test inputs

3.Expected boolean value returned

Test Case 15: test_queryAdminPersonalData

Description: Ensures the functionality of queryAdminPersonalData. When user sends the queryAdminPersonalData request ,ISCRS.queryAdminPersonalData() is invoked, it takes *token* as input and return a list of Arraylist of expected result. The list should contain only one arraylist as the admin id is unique, and the arraylist stores the following information *ID,FIRSTNAME, LASTNAME, DATEOFBIRTH, GENDER, DEPARTMENT* in order. If query is not valid or no matching results, the method returns an empty list.

Test Inputs:

1. *token(id=0,type=ADMIN,timestamp="2015.08.01.17.30.05")*
2. *token(id=0,type=BOTH,timestamp="2015.08.01.17.30.05")*
3. *token(id=0,type=STUDENT,timestamp="2015.08.01.17.30.05")*, 0 (the token type is not ADMIN or BOTH)
4. *token(id=0,type=UNDEFINED,timestamp="2015.08.01.17.30.05")*, 0 (the token type is not ADMIN or BOTH)

Expected Results:

1.For 1-2, a list of Arraylist as specified in description
2.For 3-4, empty list

Dependencies: None

Initialization: Administrator Table stored in the database

Test Step:

1.The user sends log in request and a token object generated
2.The user sends the queryAdminPersonalData request by invokeISCRS.queryAdminPersonalData() method with inputs as specified in test inputs
3.Expected results returned