المشروع النهائي (أكاديمية حسوب)

مشروع مدرسة

الاسم: المهدي بن راشد القرني

المدرب: إياد

الخوارزمية (Algorithm):

بداية البرنامج:

1. **الاتصال بقاعدة البيانات وإنشاء الجداول:**
   - الاتصال بقاعدة البيانات school.db.
   - إنشاء الجداول التالية إذا لم تكن موجودة:
     - students: لتخزين بيانات الطلاب.
     - lessons_list: لتخزين الدروس المتاحة.
     - student_lessons: كجدول وسيط لإدارة العلاقة المتعددة بين الطلاب والدروس.
2. **عرض القائمة الرئيسية:**
   - عرض الخيارات التالية:
     - **1**: إدارة الطلاب.
     - **2**: إدارة الدروس.
     - **3**: إنهاء البرنامج.

إذا كان الاختيار 1 (إدارة الطلاب):

1. **عرض قائمة إدارة الطلاب:**
   - a: إضافة طالب.
   - d: حذف طالب.
   - u: تعديل طالب.
   - s: عرض معلومات طالب.
   - b: العودة إلى القائمة الرئيسية.
2. **معالجة اختيار المستخدم:**

a: إضافة طالب

- طلب معلومات الطالب:
  - رقم الطالب (يجب أن يكون فريدًا).
  - الاسم.

- الكنية.
- العمر.
- الصف.
- تاريخ التسجيل.

- عرض قائمة الدروس المتاحة.
- طلب اختيار الدروس التي يسجل فيها الطالب (عن طريق إدخال معرف الدرس .((lesson_id
- إدخال بيانات الطالب في جدول .students
- إدخال الدروس المختارة في جدول .student_lessons
- عرض رسالة نجاح.

:dحذف طالب

- طلب رقم الطالب المراد حذفه.
- التحقق من وجود الطالب في جدول .students
- حذف جميع الدروس المرتبطة بالطالب من جدول .student_lessons
- حذف الطالب من جدول .students
- عرض رسالة نجاح.

:uتعديل طالب

- طلب رقم الطالب المراد تعديله.
- التحقق من وجود الطالب في جدول .students
- طلب البيانات الجديدة (الاسم، الكنية، العمر، الصف، تاريخ التسجيل).
- عرض قائمة الدروس المتاحة.
- طلب تحديث الدروس (استبدال الدروس القديمة بالجديدة).
- تحديث بيانات الطالب في جدول .students
- تحديث الدروس في جدول .student_lessons
- عرض رسالة نجاح.

:sعرض معلومات طالب

- طلب رقم الطالب المراد عرض معلوماته.

- التحقق من وجود الطالب في جدول .students
- استرجاع معلومات الطالب من جدول .students
- استرجاع الدروس المسجلة للطالب من جدول .student_lessons
- عرض المعلومات الكاملة للطالب (البيانات الشخصية + الدروس المسجلة).

b: العودة إلى القائمة الرئيسية

- إنهاء قائمة إدارة الطلاب والعودة إلى القائمة الرئيسية.

إذا كان الاختيار 2 (إدارة الدروس):

1. **عرض قائمة إدارة الدروس:**
   - a: إضافة درس.
   - u: تعديل درس.
   - d: حذف درس.
   - l: عرض قائمة الدروس.
   - b: العودة إلى القائمة الرئيسية.
2. **معالجة اختيار المستخدم:**

a: إضافة درس

- طلب اسم الدرس الجديد.
- إدخال الدرس في جدول .lessons_list
- عرض رسالة نجاح.

u: تعديل درس

- عرض قائمة الدروس المتاحة.
- طلب معرف الدرس (lesson_id) المراد تعديله.
- طلب الاسم الجديد للدرس.
- تحديث اسم الدرس في جدول .lessons_list
- عرض رسالة نجاح.

d: حذف درس

- عرض قائمة الدروس المتاحة.
- طلب معرف الدرس (lesson_id) المراد حذفه.
- حذف جميع السجلات المرتبطة بالدرس من جدول .student_lessons
- حذف الدرس من جدول .lessons_list
- عرض رسالة نجاح.

l:عرض قائمة الدروس

- استرجاع جميع الدروس من جدول .lessons_list
- عرضها للمستخدم.

b:العودة إلى القائمة الرئيسية

- إنهاء قائمة إدارة الدروس والعودة إلى القائمة الرئيسية.

إذا كان الاختيار 3 (إنهاء البرنامج):

1. **إغلاق الاتصال بقاعدة البيانات.**
2. **إنهاء البرنامج.**

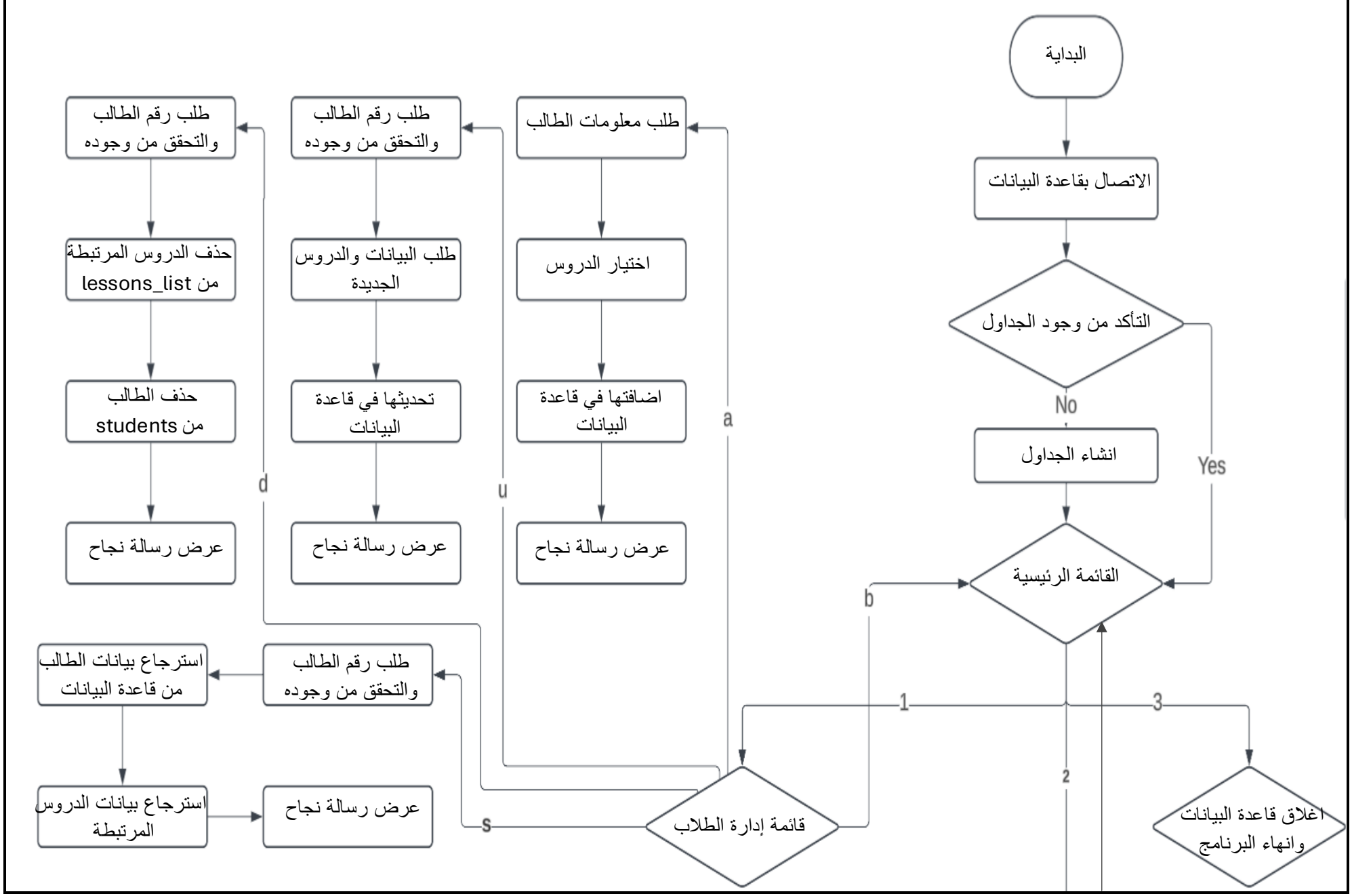
إذا كان الاختيار غير صالح:

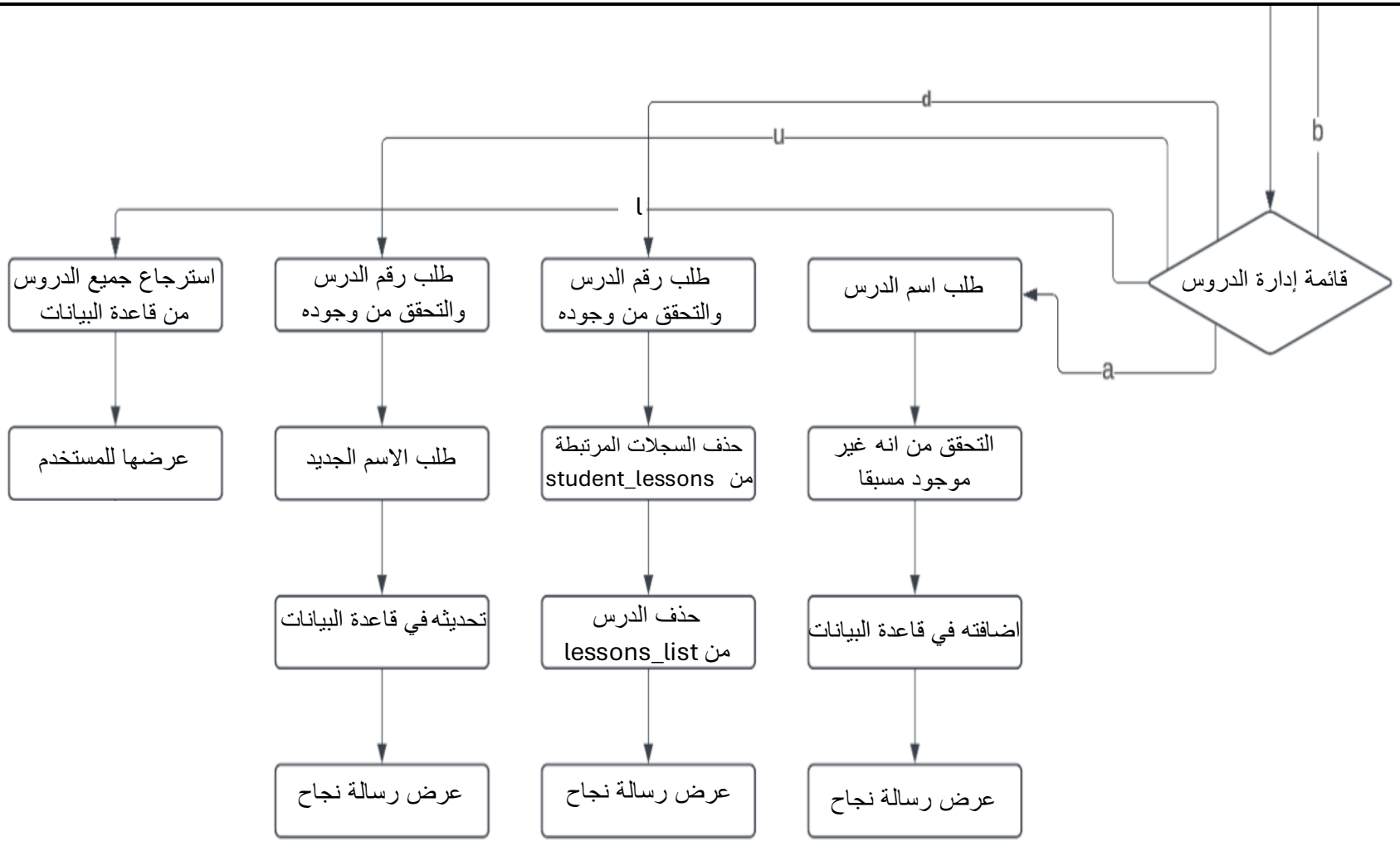
1. **عرض رسالة خطأ.**
2. **العودة إلى القائمة الرئيسية.**

إنهاء البرنامج:

1. **إغلاق الاتصال بقاعدة البيانات.**
2. **إنهاء البرنامج.**

مخطط التدفق (flowchart):




d
u
l
b
a
قائمة إدارة الدروس
طلب اسم الدرس
طلب رقم الدرس والتحقق من وجوده
طلب رقم الدرس والتحقق من وجوده
استرجاع جميع الدروس من قاعدة البيانات
التحقق من انه غير موجود مسبقا
حذف السجلات المرتبطة من student_lessons
طلب الاسم الجديد
عرضها للمستخدم
اضافته في قاعدة البيانات
حذف الدرس من lessons_list
تحديثه في قاعدة البيانات
عرض رسالة نجاح
عرض رسالة نجاح
عرض رسالة نجاح

```
CONNECT to database 'school.db'
CREATE TABLE IF NOT EXISTS students (
    student_number INTEGER PRIMARY KEY,
    name TEXT,
    surname TEXT,
    age INTEGER,
    class TEXT,
    registration_date TEXT
)

CREATE TABLE IF NOT EXISTS lessons_list (
    lesson_id INTEGER PRIMARY KEY AUTOINCREMENT,
    lesson_name TEXT UNIQUE
)

CREATE TABLE IF NOT EXISTS student_lessons (
    student_number INTEGER,
    lesson_id INTEGER,
    FOREIGN KEY(student_number) REFERENCES students(student_number),
    FOREIGN KEY(lesson_id) REFERENCES lessons_list(lesson_id),
    PRIMARY KEY(student_number, lesson_id)
)
COMMIT changes to the database
FUNCTION display_lessons():
    QUERY all lessons from lessons_list
    PRINT available lessons
```

```
FUNCTION add_lesson():
    WHILE True:
        INPUT lesson_name
        IF lesson_name is not alphabetic:
            PRINT "Invalid input, please try again"
            CONTINUE
        IF lesson_name already exists in lessons_list:
            PRINT "Lesson already exists"
            CONTINUE
        INSERT lesson_name into lessons_list
        PRINT "Lesson added successfully"
        BREAK
FUNCTION update_lesson():
    CALL display_lessons()
    WHILE True:
        INPUT lesson_id to update
        INPUT new lesson_name
        IF new lesson_name is not alphabetic:
            PRINT "Invalid input, please try again"
            CONTINUE
        UPDATE lesson_name in lessons_list WHERE lesson_id matches
        IF update successful:
            PRINT "Lesson updated successfully"
        ELSE:
            PRINT "Lesson not found"
        BREAK
FUNCTION delete_lesson():
```

```
    CALL display_lessons()
    WHILE True:
        INPUT lesson_id to delete
        DELETE from student_lessons WHERE lesson_id matches
        DELETE from lessons_list WHERE lesson_id matches
        IF deletion successful:
            PRINT "Lesson deleted successfully"
        ELSE:
            PRINT "Lesson not found"
        BREAK
FUNCTION add_student():
    WHILE True:
        INPUT student_number
        IF student_number is not 10 digits:
            PRINT "Student number must be exactly 10 digits"
            CONTINUE
        IF student_number already exists in students:
            PRINT "Student number already exists"
            CONTINUE
        BREAK

    WHILE True:
        INPUT first name
        INPUT last name
        IF first name or last name is not alphabetic:
            PRINT "Name must contain only alphabetic characters"
            CONTINUE
```

```
        BREAK

    WHILE True:
        INPUT age
        IF age is not an integer:
            PRINT "Age must be an integer"
            CONTINUE
        BREAK

    INPUT class
    INPUT registration date

    CALL display_lessons()
    WHILE True:
        INPUT lesson_id to add (or 'done' to finish)
        IF lesson_id is 'done':
            BREAK
        IF lesson_id is valid and not already assigned:
            ADD lesson_id to lessons list
        ELSE:
            PRINT "Invalid lesson ID"

    INSERT student into students table
    FOR each lesson_id in lessons list:
        INSERT into student_lessons table
    PRINT "Student added successfully"
FUNCTION delete_student():
```

```
    WHILE True:
        INPUT student_number to delete
        IF student_number is not 10 digits:
            PRINT "Student number must be exactly 10 digits"
            CONTINUE
        BREAK

    DELETE from student_lessons WHERE student_number matches
    DELETE from students WHERE student_number matches
    IF deletion successful:
        PRINT "Student deleted successfully"
    ELSE:
        PRINT "Student not found"
FUNCTION update_student():
    WHILE True:
        INPUT student_number to update
        IF student_number is not 10 digits:
            PRINT "Student number must be exactly 10 digits"
            CONTINUE
        BREAK

    QUERY student details from students table
    IF student not found:
        PRINT "Student not found"
        RETURN

    INPUT new first name (leave blank to keep current)
```

```
    INPUT new last name (leave blank to keep current)
    INPUT new age (leave blank to keep current)
    INPUT new class (leave blank to keep current)
    INPUT new registration date (leave blank to keep current)

    UPDATE student details in students table

    CALL display_lessons()
    QUERY current lessons for student
    PRINT current lessons

    WHILE True:
        INPUT lesson_id to add (or 'done' to finish)
        IF lesson_id is 'done':
            BREAK
        IF lesson_id is valid and not already assigned:
            ADD lesson_id to new lessons list
        ELSE:
            PRINT "Invalid lesson ID"

    FOR each lesson_id in new lessons list:
        INSERT into student_lessons table
    PRINT "Student updated successfully"
FUNCTION show_student():
    WHILE True:
        INPUT student_number
        QUERY student details from students table
```

```
        IF student not found:
            PRINT "Student not found"
            CONTINUE
        BREAK

    QUERY lessons for student from student_lessons and lessons_list tables
    PRINT student details and enrolled lessons
FUNCTION main_menu():
    WHILE True:
        PRINT "Main Menu:"
        PRINT "1. Student Management"
        PRINT "2. Lesson Management"
        PRINT "3. Exit"
        INPUT choice

        IF choice is '1':
            CALL student_menu()
        ELSE IF choice is '2':
            CALL lesson_menu()
        ELSE IF choice is '3':
            BREAK
        ELSE:
            PRINT "Invalid choice"
FUNCTION student_menu():
    WHILE True:
        PRINT "Student Management:"
        PRINT "a. Add Student"
```

```
        PRINT "d. Delete Student"
        PRINT "u. Update Student"
        PRINT "s. Show Student"
        PRINT "b. Back to Main Menu"
        INPUT choice

        IF choice is 'a':
            CALL add_student()
        ELSE IF choice is 'd':
            CALL delete_student()
        ELSE IF choice is 'u':
            CALL update_student()
        ELSE IF choice is 's':
            CALL show_student()
        ELSE IF choice is 'b':
            BREAK
        ELSE:
            PRINT "Invalid choice"
FUNCTION lesson_menu():
    WHILE True:
        PRINT "Lesson Management:"
        PRINT "a. Add Lesson"
        PRINT "u. Update Lesson"
        PRINT "d. Delete Lesson"
        PRINT "l. List Lessons"
        PRINT "b. Back to Main Menu"
        INPUT choice
```

```
        IF choice is 'a':
            CALL add_lesson()
        ELSE IF choice is 'u':
            CALL update_lesson()
        ELSE IF choice is 'd':
            CALL delete_lesson()
        ELSE IF choice is 'l':
            CALL display_lessons()
        ELSE IF choice is 'b':
            BREAK
        ELSE:
            PRINT "Invalid choice"
CALL main_menu()
CLOSE database connection
```