```swift
//
//  ViewController.swift
//  Smithington Public High School Library
//
//  Created by Colten Seevers & Nick Kortz on 1/29/18.
//  Copyright © 2018 Colten & Nick Kortz. All rights reserved.
//

import UIKit
import SQLite3

enum BookType: String {
    case nonfiction = "Nonfiction"
    case fiction = "Fiction"
    case historical = "Historical Nonfiction"
}

enum UserType: String {
    case admin = "Admin"
    case user = "User"
}

enum StatusType: String {
    case available = "Available"
    case out = "Out"
    case reserve = "Reserved"
}


var bookArray = [Book]()
var Accounts = [User]()
var Current = [User]()
var CurrentUser = ""
var CurrentCode = ""
var CurrentBookList = Book()
var CurrentUserType = UserType.user
var CurrentReservedList = Book()
var CurrentDue = String()
let CurrentDate = Date()
let calendar = Calendar.current
let futureDate = Date()
let year = calendar.component(.year, from: futureDate)
let month = calendar.component(.month, from: futureDate)
let day = calendar.component(.day, from: futureDate)

var dateComponent = DateComponents()
var CObook = String()
var CObarcode = String()
var COdue = String()
var RESbook = String()
var RESbarcode = String()
var category = String()
var CurrentTableID = String()
var db: OpaquePointer?
```

```
var USER: OpaquePointer?
var count = 0



class ViewController: UIViewController {


    var BookList = [Book]()


    override func viewDidLoad() {
        super.viewDidLoad()

        //Setup Check Out Period
        dateComponent.day = 7
        //Setup the SQlite3 database
        setupdatabase()
        //Load the books
        setUpBooks()
        //Update the book array with who has what checked out
        UpdateBookArray()
        //A print function that will output the current user information as it is
         entered into the databse
        GrabInfo()

        //Update the variable for the checkout status
        UpdateCurrentBookArray()
        //A print statement that will print the entire database
        readValues()

    }

    public func setupdatabase(){
        let fileURL = try! FileManager.default.url(for: .documentDirectory,
         in: .userDomainMask, appropriateFor: nil, create: false)
            .appendingPathComponent("UserDatabase.sqlite")


        if sqlite3_open(fileURL.path, &USER) != SQLITE_OK {
            print("error opening database")
        }
        else{
            print("Database Open")
        }

        if sqlite3_exec(USER, "CREATE TABLE IF NOT EXISTS USER (id INTEGER PRIMARY
         KEY AUTOINCREMENT, name TEXT, category TEXT, userID INTEGER, CObook TEXT,
         CObarcode INTEGER, COdue TEXT, RESbook TEXT, RESbarcode INTEGER)", nil,
         nil,nil) != SQLITE_OK {
            let errmsg = String(cString: sqlite3_errmsg(USER)!)
            print("error creating table: \(errmsg)")
        }
        else
```

```
    {
        print("Table Created")

    }
}
private func setUpBooks() {

    // Nonfiction
    bookArray.append(Book(name: "When Breath Becomes Air",
     category: .nonfiction, barcode:"92003",status: .available, duedate: ""))
    bookArray.append(Book(name: "Sapiens: A Brief History of Humankind",
     category: .nonfiction, barcode:"92004",status: .available, duedate: ""))
    bookArray.append(Book(name: "Into Thin Air", category: .nonfiction,
     barcode:"92005",status: .available, duedate: ""))
    bookArray.append(Book(name: "Surely You're Joking, Mr. Feynman",
     category: .nonfiction, barcode:"92006",status: .available, duedate: ""))
    bookArray.append(Book(name: "Guns, Germs, and Steel: The Fate of Human
     Societies", category: .nonfiction, barcode:"92007",status: .available,
     duedate: ""))
    bookArray.append(Book(name: "Manual for Living", category: .nonfiction,
     barcode:"92008",status: .available, duedate: ""))

    // Fiction
    bookArray.append(Book(name: "Booked", category: .fiction,
     barcode:"19208",status: .available, duedate: ""))
    bookArray.append(Book(name: "Code of Honor", category: .fiction,
     barcode:"19241",status: .available, duedate: ""))
    bookArray.append(Book(name: "Faceless", category: .fiction,
     barcode:"18909",status: .available, duedate: ""))
    bookArray.append(Book(name: "I Am Princess X", category: .fiction,
     barcode:"11046",status: .available, duedate: ""))
    bookArray.append(Book(name: "Orbiting Jupiter", category: .fiction,
     barcode:"19373",status: .available, duedate: ""))
    bookArray.append(Book(name: "Terror at Bottle Creek", category: .fiction,
     barcode:"19443",status: .available, duedate: ""))

    //Historical Nonfiction
    bookArray.append(Book(name: "The Wild Blue", category: .historical,
     barcode:"16671",status: .available, duedate: ""))
    bookArray.append(Book(name: "D-Day", category: .historical,
     barcode:"16770",status: .available, duedate: ""))
    bookArray.append(Book(name: "Citizen Soldiers: The U.S. Arm",
     category: .historical, barcode:"17720",status: .available, duedate: ""))
    bookArray.append(Book(name: "Band of Brothers", category: .historical,
     barcode:"16774",status: .available, duedate: ""))
    bookArray.append(Book(name: "Brothers In Arms", category: .historical,
     barcode:"18133",status: .available, duedate: ""))
    bookArray.append(Book(name: "Navajo Code Talkers", category: .historical,
     barcode:"18204",status: .available, duedate: ""))



}
```

```
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        //Dispose of any resources that can be recreated.
    }
}
public func readValues(){

    let queryString = "SELECT * FROM USER"

    var stmt:OpaquePointer?

    if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
        let errmsg = String(cString: sqlite3_errmsg(USER)!)
        print("error preparing insert: \(errmsg)")
        return
    }

    while(sqlite3_step(stmt) == SQLITE_ROW){
        let c0 = sqlite3_column_int(stmt, 0)
        let c1 = String(cString: sqlite3_column_text(stmt, 1))
        let c2 = String(cString: sqlite3_column_text(stmt, 2))
        let c3 = String(cString: sqlite3_column_text(stmt, 3))
        let c4 = String(cString: sqlite3_column_text(stmt, 4))
        let c5 = String(cString: sqlite3_column_text(stmt, 5))
        let c6 = String(cString: sqlite3_column_text(stmt, 6))
        let c7 = String(cString: sqlite3_column_text(stmt, 7))
        let c8 = String(cString: sqlite3_column_text(stmt, 8))


         print("c0",c0,"c1",c1,"c2",c2,"c3",c3,"c4",c4,"c5",c5,"c6",c6,"c7",c7,"c8",
         c8)

    }
}
public func UpdateBookArray(){

    let queryString = "SELECT * FROM USER"

    var stmt:OpaquePointer?

    if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
        let errmsg = String(cString: sqlite3_errmsg(USER)!)
        print("error preparing insert: \(errmsg)")
        return
    }

    while(sqlite3_step(stmt) == SQLITE_ROW){
        let c4 = String(cString: sqlite3_column_text(stmt, 4))
        let c6 = String(cString: sqlite3_column_text(stmt, 6))
        let c7 = String(cString: sqlite3_column_text(stmt, 7))

        count = 0
        for book in bookArray
        {
```

```
            if(book.name == c4)
            {
                bookArray.remove(at: count)
                bookArray.append(Book(name: book.name, category: book.category,
                 barcode: book.barcode, status: .out, duedate: c6))

            }
            count += 1
        }
        count = 0
        for book in bookArray
        {
            if(book.name == c7)
            {
                bookArray.remove(at: count)
                bookArray.append(Book(name: book.name, category: book.category,
                 barcode: book.barcode, status: .reserve, duedate: ""))
            }
            count += 1
        }
    }
}

public func UpdateCurrentBookArray(){

    let queryString = "SELECT * FROM USER"

    var stmt:OpaquePointer?

    if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
        let errmsg = String(cString: sqlite3_errmsg(USER)!)
        print("error preparing insert: \(errmsg)")
        return
    }

    while(sqlite3_step(stmt) == SQLITE_ROW){
        let c0 = sqlite3_column_int(stmt, 0)
        let c4 = String(cString: sqlite3_column_text(stmt, 4))
        let c5 = String(cString: sqlite3_column_text(stmt, 5))
        let c6 = String(cString: sqlite3_column_text(stmt, 6))
        let c7 = String(cString: sqlite3_column_text(stmt, 7))

        count = 0

        for book in bookArray
        {
            if(book.name == c4 && CurrentTableID == String(c0))
            {
                CurrentBookList = book
                CurrentDue = book.duedate
                CObook = c4
                CObarcode = c5
                COdue = c6
            }
```

```
            count += 1
        }
        count = 0
        for book in bookArray
        {
            if(book.name == c7 && CurrentTableID == String(c0))
            {
               RESbook = c7


            }
            count += 1
        }
    }
}

public func GrabInfo(){

    let queryString = "SELECT * FROM USER"

    var stmt:OpaquePointer?

    if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
        let errmsg = String(cString: sqlite3_errmsg(USER)!)
        print("error preparing insert: \(errmsg)")
        return
    }

    while(sqlite3_step(stmt) == SQLITE_ROW){
        let c0 = sqlite3_column_int(stmt, 0)
        if (String(c0) == CurrentTableID)
        {
            let c0 = sqlite3_column_int(stmt, 0)
            let c1 = String(cString: sqlite3_column_text(stmt, 1))
            let c2 = String(cString: sqlite3_column_text(stmt, 2))
            let c3 = String(cString: sqlite3_column_text(stmt, 3))
            let c4 = String(cString: sqlite3_column_text(stmt, 4))
            let c5 = String(cString: sqlite3_column_text(stmt, 5))
            let c6 = String(cString: sqlite3_column_text(stmt, 6))
            let c7 = String(cString: sqlite3_column_text(stmt, 7))
            let c8 = String(cString: sqlite3_column_text(stmt, 8))


             print("c0",c0,"c1",c1,"c2",c2,"c3",c3,"c4",c4,"c5",c5,"c6",c6,"c7",c7,"
             c8",c8)

            CurrentUser = c1
            CurrentCode = c3

        }

    }

}
```

```
public func TableCheckout( CurrentCode:String, CObook:String, CObarcode:String,
 COdue:String){


     print(CurrentUser,category,CurrentCode,CObook,CObarcode,COdue,RESbook,RESbarcod
     e)

    var updateStatementStringC4 = "UPDATE USER SET CObook = '"
    updateStatementStringC4.append(CObook)
    updateStatementStringC4.append("' WHERE Id = ")
    updateStatementStringC4.append(CurrentTableID)
    updateStatementStringC4.append(";")
    var updateStatementStringC5 = "UPDATE USER SET CObarcode = '"
    updateStatementStringC5.append(CObarcode)
    updateStatementStringC5.append("' WHERE Id = ")
    updateStatementStringC5.append(CurrentTableID)
    updateStatementStringC5.append(";")
    var updateStatementStringC6 = "UPDATE USER SET COdue = '"
    updateStatementStringC6.append(COdue)
    updateStatementStringC6.append("' WHERE Id = ")
    updateStatementStringC6.append(CurrentTableID)
    updateStatementStringC6.append(";")

    print(updateStatementStringC4)
    print(updateStatementStringC5)
    print(updateStatementStringC6)
    var updateStatement: OpaquePointer? = nil

    if sqlite3_prepare_v2(USER, updateStatementStringC4, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C4 could not be prepared")
    }
    if sqlite3_prepare_v2(USER, updateStatementStringC5, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C5 could not be prepared")
    }
    if sqlite3_prepare_v2(USER, updateStatementStringC6, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
```

```
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C6 could not be prepared")
    }
    sqlite3_finalize(updateStatement)


}
public func TableCheckIN( CurrentCode:String){


     print(CurrentUser,category,CurrentCode,CObook,CObarcode,COdue,RESbook,RESbarcod
     e)
    CObook = ""
    CObarcode = ""
    COdue = ""

    var updateStatementStringC4 = "UPDATE USER SET CObook = '"
    updateStatementStringC4.append(CObook)
    updateStatementStringC4.append("' WHERE Id = ")
    updateStatementStringC4.append(CurrentTableID)
    updateStatementStringC4.append(";")
    var updateStatementStringC5 = "UPDATE USER SET CObarcode = '"
    updateStatementStringC5.append(CObarcode)
    updateStatementStringC5.append("' WHERE Id = ")
    updateStatementStringC5.append(CurrentTableID)
    updateStatementStringC5.append(";")
    var updateStatementStringC6 = "UPDATE USER SET COdue = '"
    updateStatementStringC6.append(COdue)
    updateStatementStringC6.append("' WHERE Id = ")
    updateStatementStringC6.append(CurrentTableID)
    updateStatementStringC6.append(";")

    print(updateStatementStringC4)
    print(updateStatementStringC5)
    print(updateStatementStringC6)
    var updateStatement: OpaquePointer? = nil

    if sqlite3_prepare_v2(USER, updateStatementStringC4, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C4 could not be prepared")
    }
    if sqlite3_prepare_v2(USER, updateStatementStringC5, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
```

```
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C5 could not be prepared")
    }
    if sqlite3_prepare_v2(USER, updateStatementStringC6, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C6 could not be prepared")
    }
    sqlite3_finalize(updateStatement)

}

public func TableReserve( CurrentCode:String, RESbook:String, RESbarcode:String){


     print(CurrentUser,category,CurrentCode,CObook,CObarcode,COdue,RESbook,RESbarcod
     e)

    var updateStatementStringC7 = "UPDATE USER SET RESbook = '"
    updateStatementStringC7.append(RESbook)
    updateStatementStringC7.append("' WHERE Id = ")
    updateStatementStringC7.append(CurrentTableID)
    updateStatementStringC7.append(";")
    var updateStatementStringC8 = "UPDATE USER SET RESbarcode = '"
    updateStatementStringC8.append(RESbarcode)
    updateStatementStringC8.append("' WHERE Id = ")
    updateStatementStringC8.append(CurrentTableID)
    updateStatementStringC8.append(";")

    print(updateStatementStringC7)
    print(updateStatementStringC8)
    var updateStatement: OpaquePointer? = nil

    if sqlite3_prepare_v2(USER, updateStatementStringC7, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C7 could not be prepared")
    }
    if sqlite3_prepare_v2(USER, updateStatementStringC8, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
```

```
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C8 could not be prepared")
    }
    sqlite3_finalize(updateStatement)

}
public func TableUNReserve( CurrentCode:String){


     print(CurrentUser,category,CurrentCode,CObook,CObarcode,COdue,RESbook,RESbarcod
     e)
    RESbook = ""
    RESbarcode = ""
    var updateStatementStringC7 = "UPDATE USER SET RESbook = '"
    updateStatementStringC7.append(RESbook)
    updateStatementStringC7.append("' WHERE Id = ")
    updateStatementStringC7.append(CurrentTableID)
    updateStatementStringC7.append(";")
    var updateStatementStringC8 = "UPDATE USER SET RESbarcode = '"
    updateStatementStringC8.append(RESbarcode)
    updateStatementStringC8.append("' WHERE Id = ")
    updateStatementStringC8.append(CurrentTableID)
    updateStatementStringC8.append(";")

    print(updateStatementStringC7)
    print(updateStatementStringC8)
    var updateStatement: OpaquePointer? = nil

    if sqlite3_prepare_v2(USER, updateStatementStringC7, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C7 could not be prepared")
    }
    if sqlite3_prepare_v2(USER, updateStatementStringC8, -1, &updateStatement, nil)
     == SQLITE_OK {
        if sqlite3_step(updateStatement) == SQLITE_DONE {
            print("Successfully updated row.")
        } else {
            print("Could not update row.")
        }
    } else {
        print("UPDATE statement C8 could not be prepared")
    }
    sqlite3_finalize(updateStatement)

}
```

```
public func DatabaseSetupUsers(ID:String, User:String, Category:String, Code:String,
 CObook:String, CObarcode:String, COdue:String, RESbook: String,RESbarcode:String){

    print(ID,User,category,Code,CObook,CObarcode,COdue,RESbook,RESbarcode)

    let queryString = "SELECT * FROM USER"

    var stmt:OpaquePointer?

    if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
        let errmsg = String(cString: sqlite3_errmsg(USER)!)
        print("error preparing insert: \(errmsg)")
        return
    }
        var updateStatementStringC1 = "UPDATE USER SET Name = '"
        updateStatementStringC1.append(User)
        updateStatementStringC1.append("' WHERE Id = ")
        updateStatementStringC1.append(String(ID))
        updateStatementStringC1.append(";")
        var updateStatementStringC2 = "UPDATE USER SET Category = '"
        updateStatementStringC2.append(String(Category))
        updateStatementStringC2.append("' WHERE Id = ")
        updateStatementStringC2.append(String(ID))
        updateStatementStringC2.append(";")
        var updateStatementStringC3 = "UPDATE USER SET userID = '"
        updateStatementStringC3.append(Code)
        updateStatementStringC3.append("' WHERE Id = ")
        updateStatementStringC3.append(String(ID))
        updateStatementStringC3.append(";")
        var updateStatementStringC4 = "UPDATE USER SET CObook = '"
        updateStatementStringC4.append(CObook)
        updateStatementStringC4.append("' WHERE Id = ")
        updateStatementStringC4.append(String(ID))
        updateStatementStringC4.append(";")
        var updateStatementStringC5 = "UPDATE USER SET CObarcode = '"
        updateStatementStringC5.append(CObarcode)
        updateStatementStringC5.append("' WHERE Id = ")
        updateStatementStringC5.append(String(ID))
        updateStatementStringC5.append(";")
        var updateStatementStringC6 = "UPDATE USER SET COdue = '"
        updateStatementStringC6.append(COdue)
        updateStatementStringC6.append("' WHERE Id = ")
        updateStatementStringC6.append(String(ID))
        updateStatementStringC6.append(";")
        var updateStatementStringC7 = "UPDATE USER SET RESbook = '"
        updateStatementStringC7.append(RESbook)
        updateStatementStringC7.append("' WHERE Id = ")
        updateStatementStringC7.append(String(ID))
        updateStatementStringC7.append(";")
        var updateStatementStringC8 = "UPDATE USER SET RESbarcode = '"
        updateStatementStringC8.append(RESbarcode)
        updateStatementStringC8.append("' WHERE Id = ")
        updateStatementStringC8.append(String(ID))
        updateStatementStringC8.append(";")
```

```
print(updateStatementStringC1)
print(updateStatementStringC2)
print(updateStatementStringC3)
print(updateStatementStringC4)
print(updateStatementStringC5)
print(updateStatementStringC6)

print(updateStatementStringC7)
print(updateStatementStringC8)

var updateStatement: OpaquePointer? = nil

if sqlite3_prepare_v2(USER, updateStatementStringC1, -1, &updateStatement,
 nil) == SQLITE_OK {
    if sqlite3_step(updateStatement) == SQLITE_DONE {
        print("Successfully updated row.")
    } else {
        print("Could not update row.")
    }
} else {
    print("UPDATE statement C1 could not be prepared")
}

if sqlite3_prepare_v2(USER, updateStatementStringC2, -1, &updateStatement,
 nil) == SQLITE_OK {
    if sqlite3_step(updateStatement) == SQLITE_DONE {
        print("Successfully updated row.")
    } else {
        print("Could not update row.")
    }
} else {
    print("UPDATE statement C2 could not be prepared")
}

if sqlite3_prepare_v2(USER, updateStatementStringC3, -1, &updateStatement,
 nil) == SQLITE_OK {
    if sqlite3_step(updateStatement) == SQLITE_DONE {
        print("Successfully updated row.")
    } else {
        print("Could not update row.")
    }
} else {
    print("UPDATE statement C3 could not be prepared")
}


if sqlite3_prepare_v2(USER, updateStatementStringC4, -1, &updateStatement,
 nil) == SQLITE_OK {
    if sqlite3_step(updateStatement) == SQLITE_DONE {
        print("Successfully updated row.")
    } else {
        print("Could not update row.")
    }
```

```
        } else {
            print("UPDATE statement C4 could not be prepared")
        }
        if sqlite3_prepare_v2(USER, updateStatementStringC5, -1, &updateStatement,
         nil) == SQLITE_OK {
            if sqlite3_step(updateStatement) == SQLITE_DONE {
                print("Successfully updated row.")
            } else {
                print("Could not update row.")
            }
        } else {
            print("UPDATE statement C5 could not be prepared")
        }
        if sqlite3_prepare_v2(USER, updateStatementStringC6, -1, &updateStatement,
         nil) == SQLITE_OK {
            if sqlite3_step(updateStatement) == SQLITE_DONE {
                print("Successfully updated row.")
            } else {
                print("Could not update row.")
            }
        } else {
            print("UPDATE statement C6 could not be prepared")
        }

        if sqlite3_prepare_v2(USER, updateStatementStringC7, -1, &updateStatement,
         nil) == SQLITE_OK {
            if sqlite3_step(updateStatement) == SQLITE_DONE {
                print("Successfully updated row.")
            } else {
                print("Could not update row.")
            }
        } else {
            print("UPDATE statement C7 could not be prepared")
        }
        if sqlite3_prepare_v2(USER, updateStatementStringC8, -1, &updateStatement,
         nil) == SQLITE_OK {
            if sqlite3_step(updateStatement) == SQLITE_DONE {
                print("Successfully updated row.")
            } else {
                print("Could not update row.")
            }
        } else {
            print("UPDATE statement C8 could not be prepared")
        }
        sqlite3_finalize(updateStatement)

}


// Login Expansion
func setUpUsers() {
    // USERS
    //Accounts.append(User(name: "Amber", category: .user, Code:"1002", bookList:
     [],reserveList: []))
```

```
    DatabaseSetupUsers(ID: "2", User: "Amber", Category: "User", Code: "1002",
     CObook:"", CObarcode: "",COdue: "", RESbook: "",RESbarcode: "")
    //Accounts.append(User(name: "James", category: .user, Code:"1004", bookList:
     [],reserveList: []))
    DatabaseSetupUsers(ID: "3", User: "James", Category: "User", Code: "1004",
     CObook: "", CObarcode: "", COdue: "", RESbook: "", RESbarcode: "")
    //Accounts.append(User(name: "Peter", category: .user, Code:"1006", bookList:
     [],reserveList: []))
    DatabaseSetupUsers(ID: "4", User: "Peter", Category: "User", Code: "1006",
     CObook:"", CObarcode: "",COdue: "", RESbook: "",RESbarcode: "")
    //Accounts.append(User(name: "Haywood", category: .user, Code:"1008", bookList:
     [],reserveList: []))
    DatabaseSetupUsers(ID: "5", User: "Haywood", Category: "User", Code: "1008",
     CObook:"", CObarcode: "",COdue: "", RESbook: "",RESbarcode: "")
    //Accounts.append(User(name: "James", category: .user, Code:"1010", bookList:
     [],reserveList: []))
    DatabaseSetupUsers(ID: "6", User: "Jacob", Category: "User", Code: "1010",
     CObook:"", CObarcode: "",COdue: "", RESbook: "",RESbarcode: "")
    //Accounts.append(User(name: "Shell", category: .user, Code:"1012", bookList:
     [],reserveList: []))
    DatabaseSetupUsers(ID: "7", User: "Shell", Category: "User", Code: "1012",
     CObook:"", CObarcode: "",COdue: "", RESbook: "",RESbarcode: "")
    //ADMINS
    //Accounts.append(User(name: "Admin", category: .admin, Code:"0001", bookList:
     [],reserveList: []))
    DatabaseSetupUsers(ID: "1", User: "Admin", Category: "Admin", Code: "0001",
     CObook:"", CObarcode: "",COdue: "", RESbook: "",RESbarcode: "")

}
public func DeleteUserEntry(){

    let queryString = "SELECT * FROM USER"

    var stmt:OpaquePointer?

    if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
        let errmsg = String(cString: sqlite3_errmsg(USER)!)
        print("error preparing insert: \(errmsg)")
        return
    }

    while(sqlite3_step(stmt) == SQLITE_ROW){
        var queryString = "DELETE FROM USER WHERE Id = "
        queryString.append(String(SQLITE_ROW))
        queryString.append(";")

        print(queryString)
        if sqlite3_prepare(USER, queryString, -1, &stmt, nil) != SQLITE_OK{
            if sqlite3_step(stmt) == SQLITE_DONE {
                print("Successfully deleted row.")
            } else {
                print("Could not delete row.")
            }
        } else {
```

```
            print("DELETE statement could not be prepared")
        }
        sqlite3_finalize(stmt)
    }
}
```