Program

connection.py

```
db_host_name="www.papademas.net"
db_name="pydb"
db_user="root"
db_password="jamesp"
db_table_name="brady"
```

contacts.py

```
contact_list = [
  ['Siemens, Harper',  '323-4149'],
  ['Smith, Patti',  '239-1212'],
  ['Jackson, Janet',   '313-1352'],
  ['Manfredi, Ralph','872-2221'],
  ['Thompson, Bobby',   '365-2622'],
  ['James, Lebron',  '457-6223'],
  ['Ziegler, Zig',   '667-1101'],
  ['Robbins, Tony',     '329-2310']
]
```

myDatabasefile.py

```
# coding=utf-8
#################################################################
# PROJECT:
# Working with Tkinter (part 2) & a MySQL database
# -------------------------------------------------------------
# OBJECTIVE:
# To modify your lab 6 program to work with a MySQL
# database you will create.
# -------------------------------------------------------------
# DESCRIPTION:
# This project will have you create a myDatabase.py file
# to interact with your existing tkContacts.py file you
# modified for lab 6. Your myDatabasefile.py will include
# the following functions to perform the following CRUD
# operations:
# a.        Create a table
# b.        Read from the table
# c.        Update the table
# d.        Delete from the table
# e.        Insert into the table
# -------------------------------------------------------------
# AUTHOR:       Brady Houseknecht
# BORN ON:      07/27/2014
# DUE ON:       07/28/2014 6 PM
# REVISION:     1.0
#
# -------------------------------------------------------------
# REVISION HISTORY:
# 1.0 Baseline - BH
#
################################################################

import MySQLdb
from contacts import *







class myDatabaseFile:

    def __init__(self, host, db, user, password, table_name) :
        self.db_host_name = host
        self.db_name = db
        self.db_user = user
        self.db_password = password
        self.db_table_name = table_name


    def create_connection (self) :
        db = MySQLdb.connect( self.db_host_name,self.db_user,self.db_password,self.db_name )
        return db


    def create_table (self) :
        db = self.create_connection()
        cursor = db.cursor()
        sql = "SHOW TABLES LIKE '" + self.db_table_name + "'"
        print sql
        cursor.execute(sql)
        result = cursor.fetchone()
        if not result:
            sql = "CREATE TABLE " + self.db_table_name + " (ID INT NOT NULL, NAME  CHAR(100) NOT NULL, PHONE CHAR(100), UNIQUE(ID) )"
            print sql
            try:
                cursor.execute(sql)
                self.load_table(db)
            except:
                db.rollback()
        db.close()


    def load_table (self, db) :
        global contact_list
        index = 1
        for name,phone in contact_list :
            sql = "INSERT INTO " + self.db_table_name + "(ID, NAME, PHONE) VALUES (" + str(index) + ", '"+ name +"', '" + phone + "')"
            index+=1
            print sql
            try:
                db.cursor().execute(sql)
                db.commit()
            except:
                db.rollback()


    def read_table (self) :
        contacts = []
        db = self.create_connection()
        cursor = db.cursor()
        sql = "SELECT NAME, PHONE FROM " + self.db_table_name + " ORDER BY NAME ASC"
        print sql
        try:
            cursor.execute(sql)
            results = cursor.fetchall()
            for row in results:
                name = row[0]
                phone = row[1]
                contacts.append([name, phone])

        except:
            print "Error: unable to read the " + self.db_table_name + " table."

        db.close()
        return contacts


    def read_table_next_id (self) :
        next_id = 0
        db = self.create_connection()
        cursor = db.cursor()
        sql = "SELECT MAX(ID)+1 'NEXT_ID' FROM " + self.db_table_name
        print sql
        try:
            cursor.execute(sql)
            result = cursor.fetchone()
            next_id = int(result[0])
        except:
            db.rollback()
        db.close()
        return next_id


    def read_table_valid_id (self, id) :
        rc = True
        db = self.create_connection()
        cursor = db.cursor()
        sql = "SELECT COUNT(*) 'MATCHES' FROM " + self.db_table_name + " WHERE ID = " + str(id)
        print sql
        try:
            cursor.execute(sql)
            result = cursor.fetchone()
            if (int(result[0]) == 0):
                rc = False
        except:
            db.rollback()
            rc = False
        db.close()
        return rc


    def read_table_max_id (self, name) :
        max_id = 0
        db = self.create_connection()
        cursor = db.cursor()
        sql = "SELECT MAX(ID) 'ID' FROM " + self.db_table_name + " WHERE NAME='" + name + "'"
        print sql
        try:
            cursor.execute(sql)
            result = cursor.fetchone()
            max_id = int(result[0])
        except:
            db.rollback()
        db.close()
        return max_id


    def insert_table (self, name, phone) :
        rc = True
        next_id = self.read_table_next_id()
        existing_id = self.read_table_max_id(name)
        if(next_id > 0 & existing_id == 0):
            db = self.create_connection()
            cursor = db.cursor()
            sql = "INSERT INTO " + self.db_table_name + "(ID, NAME, PHONE) VALUES ("+ str(next_id) +", '"+ name +"', '" + phone + "')"
            print sql
            try:
                db.cursor().execute(sql)
                db.commit()
            except:
                db.rollback()
                rc = False
            db.close()
        else:
            rc = False
        return rc


    def update_table (self, id, name, phone) :
        rc = True
        if(self.read_table_valid_id(id)):
            db = self.create_connection()
            cursor = db.cursor()
            sql = "UPDATE " + self.db_table_name + " SET NAME ='" + name + "', PHONE = '" + phone + "' WHERE ID = " + str(id)
            print sql
            try:
                db.cursor().execute(sql)
                db.commit()
            except:
                db.rollback()
                rc = False
            db.close()
        else:
            rc = False
        return rc


    def delete_table (self, id) :
        rc = True
        if(self.read_table_valid_id(id)):
            db = self.create_connection()
            cursor = db.cursor()
            sql = "DELETE FROM " + self.db_table_name + " WHERE ID = " + str(id)
            print sql
            try:
                db.cursor().execute(sql)
                db.commit()
            except:
                db.rollback()
                rc = False
            db.close()
        else:
            rc = False
        return rc
```

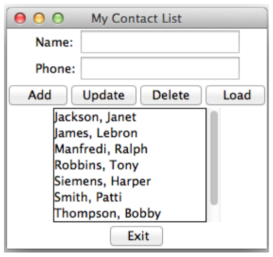
tkContacts.py

```
# coding=utf-8
#################################################################
# PROJECT:
# Final Lab - Working with Tkinter (part 2) & a MySQL database
# --------------------------------------------------------------
# OBJECTIVE:
# To modify your lab 6 program to work with a MySQL
# database you will create.
# --------------------------------------------------------------
# DESCRIPTION:
# 1.  Adjust any functions you see fit from your tkContacts.py
# script, so that any updates, deletes, loads, or adds are
# performed by the said operations defined in your
# myDatabasefile.py script. Note here you don't really need a
# 'Save' button, unless you deem it worthy to have it somehow,
# so just delete it from the GUI and any respective callback
# function defined that's glued to it.
# You see when the user presses your button, your callback
# function should automatically get passed the right contact
# information selected by the user which in turn will make the
# necessary changes to the particular contact record on the
# server immediately!
#
# 2. Keep any remaining functions you deem necessary to
# have the correct running app, such as makeWindow(),
# setSelect() and whichSelected(). Your program should
# ultimately load in all the records your inserted in
# step 1 into the listbox, similarly to how you had
# the records load into the listbox from our contacts.py
# file in lab 6.
# --------------------------------------------------------------
# AUTHOR:       Brady Houseknecht
# BORN ON:      07/27/2014
# DUE ON:       07/28/2014 6 PM
# REVISION:     1.0
#
# --------------------------------------------------------------
# REVISION HISTORY:
# 1.0 Baseline - BH
#
################################################################

import os
from Tkinter import *
import tkMessageBox
from connection import *
from myDatabasefile import *

app_title="My Contact List"

db = myDatabaseFile(db_host_name, db_name, db_user, db_password, db_table_name)

app_contact_id = -1

def clearContactId () :
    global app_contact_id
    app_contact_id = -1

def setContactId (name) :
    global app_contact_id
    app_contact_id = db.read_table_max_id(name)
```

```
def getContactId () :
    global app_contact_id
    return app_contact_id

def selection () :
    try:
        return int(select.curselection()[0])
    except:
        return -1


def onAddContact () :
    if(nameVar.get() == ""):
        showError("Please enter a valid name for the new contact")
    else:
        addContact()


def addContact () :
    if (db.insert_table(nameVar.get(), phoneVar.get())):
        setList ()
    else:
        showError("Please make sure the name that does not already exist.")


def showError (msg) :
    global app_title
    tkMessageBox.showerror(title=app_title + ":~ Error", \
    message=msg)


def onUpdateContact () :
    name = nameVar.get()
    phone = phoneVar.get()
    if (name == "" ):
        showError("The name field cannot be left blank. Please enter a value.")
    else:
        updateContact()


def updateContact () :
    if (db.update_table(getContactId(), nameVar.get(), phoneVar.get())):
        setList ()
    else:
        showError("Failed to update database.  Please try again.")


def onDeleteContact () :
    if(selection() > 0):
        name, phone = contactlist[selection()]
        deleteContact(name)
    elif(nameVar.get()!=""):
        deleteContact(nameVar.get())
    else:
        showError("Please select or load a contact before pressing delete.")


def deleteContact (name) :
    global app_title
    if (tkMessageBox.askokcancel(title=app_title, \
        message="Are you sure want to delete " + name +"?") == 1):
        if (db.delete_table(getContactId())):
            nameVar.set("")
            phoneVar.set("")
            clearContactId()
            setList ()
        else:
            showError("Failed to update database.  Please try again.")


def loadContact  () :
    name, phone = contactlist[selection()]
    setContactId(name)
    nameVar.set(name)
    phoneVar.set(phone)


def confirmExit () :
    global app_title
    if (tkMessageBox.askokcancel(title=app_title, \
        message="Are you want to exit?") == 1):
            os._exit(1)


def buildFrame () :
    global nameVar, phoneVar, select, app_title
    root = Tk()

    frame1 = Frame(root)
    frame1.master.title(app_title)
    frame1.pack()

    Label(frame1, text="Name:").grid(row=0, column=0, sticky=W)
    nameVar = StringVar()
    name = Entry(frame1, textvariable=nameVar)
    name.grid(row=0, column=1, sticky=W)

    Label(frame1, text="Phone:").grid(row=1, column=0, sticky=W)
    phoneVar= StringVar()
    phone= Entry(frame1, textvariable=phoneVar)
    phone.grid(row=1, column=1, sticky=W)

    frame1 = Frame(root)       # add a row of buttons
    frame1.pack()
    btn1 = Button(frame1,text=" Add  ",command=onAddContact)
    btn2 = Button(frame1,text="Update",command=onUpdateContact)
    btn3 = Button(frame1,text="Delete",command=onDeleteContact)
    btn4 = Button(frame1,text=" Load ",command=loadContact)
    btn1.pack(side=LEFT); btn2.pack(side=LEFT)
    btn3.pack(side=LEFT); btn4.pack(side=LEFT)

    frame1 = Frame(root)       # allow for selection of names
    frame1.pack()
    scroll = Scrollbar(frame1, orient=VERTICAL)
    select = Listbox(frame1, yscrollcommand=scroll.set, height=7)
    scroll.config (command=select.yview)
    scroll.pack(side=RIGHT, fill=Y)
    select.pack(side=LEFT,  fill=BOTH)

    frame1 = Frame(root)    # add an Exit button at the bottom
    frame1.pack()
    btn6 = Button(frame1,text=" Exit ",command=confirmExit)
    btn6.pack(side=BOTTOM)

    return root


def setList () :
    global contactlist
    contactlist = db.read_table()
    select.delete(0,END)
    for name,phone in contactlist :
        select.insert (END, name)


root = buildFrame()
db.create_table()
setList ()
root.mainloop()
os._exit(1)
```

Output

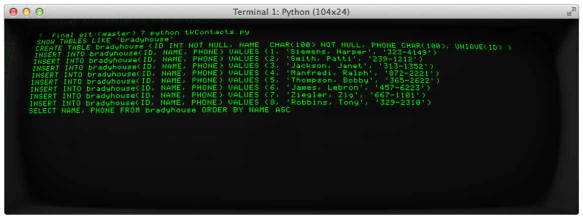
Startup



Name:
Phone:

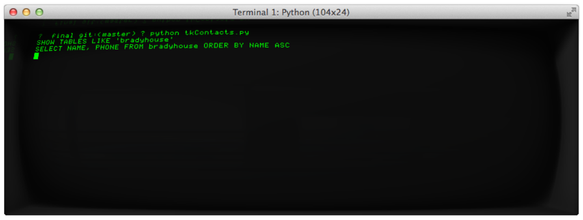
Jackson, Janet
James, Lebron
Manfredi, Ralph
Robbins, Tony
Siemens, Harper
Smith, Patti
Thompson, Bobby

First Time (Console Output)



```
? final git:(master) ? python tkContacts.py
SHOW TABLES LIKE 'bradyhouse'
CREATE TABLE bradyhouse (ID INT NOT NULL, NAME  CHAR(100) NOT NULL, PHONE CHAR(100), UNIQUE(ID) )
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (1, 'Siemens, Harper', '323-4149')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (2, 'Smith, Patti', '239-1212')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (3, 'Jackson, Janet', '313-1352')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (4, 'Manfredi, Ralph', '872-2221')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (5, 'Thompson, Bobby', '365-2622')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (6, 'James, Lebron', '457-6223')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (7, 'Ziegler, Zig', '667-1101')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (8, 'Robbins, Tony', '329-2310')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
```

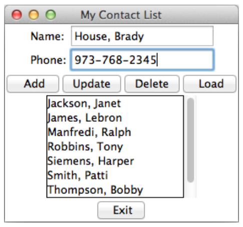
N Time (Console Output)



```
? final git:(master) ? python tkContacts.py
SHOW TABLES LIKE 'bradyhouse'
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
```

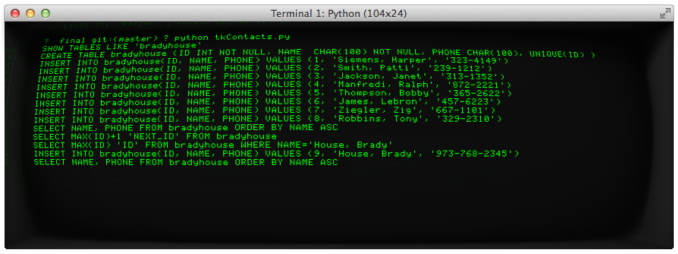
Add Contact

Click Add (Console Output)



Name: House, Brady
Phone: 973-768-2345

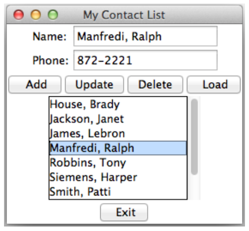
Jackson, Janet
James, Lebron
Manfredi, Ralph
Robbins, Tony
Siemens, Harper
Smith, Patti
Thompson, Bobby



```
? final git:(master) ? python tkContacts.py
SHOW TABLES LIKE 'bradyhouse'
CREATE TABLE bradyhouse (ID INT NOT NULL, NAME  CHAR(100) NOT NULL, PHONE CHAR(100), UNIQUE(ID) )
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (1, 'Siemens, Harper', '323-4149')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (2, 'Smith, Patti', '239-1212')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (3, 'Jackson, Janet', '313-1352')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (4, 'Manfredi, Ralph', '872-2221')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (5, 'Thompson, Bobby', '365-2622')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (6, 'James, Lebron', '457-6223')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (7, 'Ziegler, Zig', '667-1101')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (8, 'Robbins, Tony', '329-2310')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID)+1 'NEXT_ID' FROM bradyhouse
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='House, Brady'
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (9, 'House, Brady', '973-768-2345')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
```

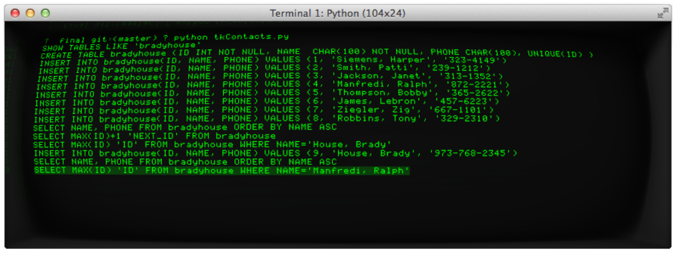
Load Contact

Click Load (Console Output)



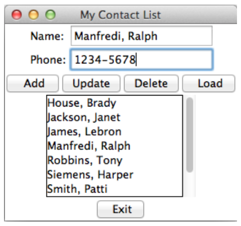
Name: Manfredi, Ralph
Phone: 872-2221

House, Brady
Jackson, Janet
James, Lebron
Manfredi, Ralph
Robbins, Tony
Siemens, Harper
Smith, Patti



```
? final git:(master) ? python tkContacts.py
SHOW TABLES LIKE 'bradyhouse'
CREATE TABLE bradyhouse (ID INT NOT NULL, NAME  CHAR(100) NOT NULL, PHONE CHAR(100), UNIQUE(ID) )
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (1, 'Siemens, Harper', '323-4149')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (2, 'Smith, Patti', '239-1212')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (3, 'Jackson, Janet', '313-1352')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (4, 'Manfredi, Ralph', '872-2221')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (5, 'Thompson, Bobby', '365-2622')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (6, 'James, Lebron', '457-6223')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (7, 'Ziegler, Zig', '667-1101')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (8, 'Robbins, Tony', '329-2310')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID)+1 'NEXT_ID' FROM bradyhouse
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='House, Brady'
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (9, 'House, Brady', '973-768-2345')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='Manfredi, Ralph'
```

Update Contact

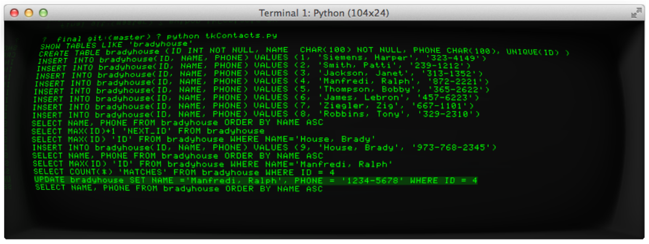

My Contact List

Name: Manfredi, Ralph

Phone: 1234-5678

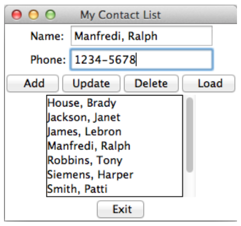
```
House, Brady
Jackson, Janet
James, Lebron
Manfredi, Ralph
Robbins, Tony
Siemens, Harper
Smith, Patti
```

Click Update (Console Output)


Terminal 1: Python (104x24)

```
? final git:(master) ? python tkContacts.py
SHOW TABLES LIKE 'bradyhouse'
CREATE TABLE bradyhouse (ID INT NOT NULL, NAME  CHAR(100) NOT NULL, PHONE CHAR(100), UNIQUE(ID) )
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (1, 'Siemens, Harper', '323-4149')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (2, 'Smith, Patti', '239-1212')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (3, 'Jackson, Janet', '313-1352')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (4, 'Manfredi, Ralph', '872-2221')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (5, 'Thompson, Bobby', '365-2622')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (6, 'James, Lebron', '457-6223')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (7, 'Ziegler, Zig', '667-1101')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (8, 'Robbins, Tony', '329-2310')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID)+1 'NEXT_ID' FROM bradyhouse
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='House, Brady'
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (9, 'House, Brady', '973-768-2345')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='Manfredi, Ralph'
SELECT COUNT(*) 'MATCHES' FROM bradyhouse WHERE ID = 4
UPDATE bradyhouse SET NAME ='Manfredi, Ralph', PHONE = '1234-5678' WHERE ID = 4
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
```

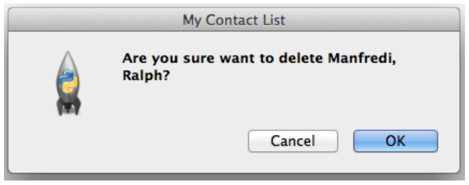
Delete Contact


My Contact List

Name: Manfredi, Ralph

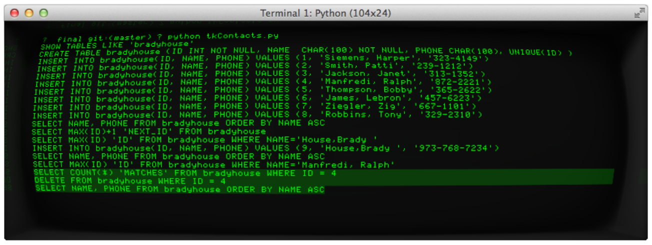
Phone: 1234-5678

```
House, Brady
Jackson, Janet
James, Lebron
Manfredi, Ralph
Robbins, Tony
Siemens, Harper
Smith, Patti
```


My Contact List

Are you sure want to delete Manfredi, Ralph?

Click Delete(Console Output)


Terminal 1: Python (104x24)

```
? final git:(master) ? python tkContacts.py
SHOW TABLES LIKE 'bradyhouse'
CREATE TABLE bradyhouse (ID INT NOT NULL, NAME  CHAR(100) NOT NULL, PHONE CHAR(100), UNIQUE(ID) )
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (1, 'Siemens, Harper', '323-4149')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (2, 'Smith, Patti', '239-1212')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (3, 'Jackson, Janet', '313-1352')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (4, 'Manfredi, Ralph', '872-2221')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (5, 'Thompson, Bobby', '365-2622')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (6, 'James, Lebron', '457-6223')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (7, 'Ziegler, Zig', '667-1101')
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (8, 'Robbins, Tony', '329-2310')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID)+1 'NEXT_ID' FROM bradyhouse
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='House,Brady '
INSERT INTO bradyhouse(ID, NAME, PHONE) VALUES (9, 'House,Brady ', '973-768-7234')
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
SELECT MAX(ID) 'ID' FROM bradyhouse WHERE NAME='Manfredi, Ralph'
SELECT COUNT(*) 'MATCHES' FROM bradyhouse WHERE ID = 4
DELETE FROM bradyhouse WHERE ID = 4
SELECT NAME, PHONE FROM bradyhouse ORDER BY NAME ASC
```