NOTICE:
The copyright law of the United States (Title 17, United States Code) governs the making of reproductions of copyrighted material. One specified condition is that the reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a user makes a request for, or later uses a reproduction for purposes in excess of "fair use," that user may be liable for copyright infringement.

RESTRICTIONS:
This student work may be read, quoted from, cited, for purposes of research. It may not be published in full except by permission of the author. Per request from author do not reproduce this work in print.
MySQL database with Web Front-End and Security Features

Presented to the faculty of Lycoming College
in partial fulfillment of the requirements
for departmental Honors in Computer Science

by
Joshua P. Speicher
Lycoming College
April 24, 2002

Dr. Bahram Golshan
Dr. Eileen Peluso
Dr. David Wolfe
Mr. Dave Heffner
# Table of Contents
## For Project Documentation

<table>
<thead>
<tr>
<th>Title of Section</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Honors Proposal</td>
<td>2</td>
</tr>
<tr>
<td>Requirements Document</td>
<td>7</td>
</tr>
<tr>
<td>Script Design Pages</td>
<td></td>
</tr>
<tr>
<td>alumni.cgi</td>
<td>11</td>
</tr>
<tr>
<td>forgot.cgi</td>
<td>13</td>
</tr>
<tr>
<td>pass.cgi</td>
<td>14</td>
</tr>
<tr>
<td>myinfo.cgi</td>
<td>15</td>
</tr>
<tr>
<td>regit.cgi</td>
<td>16</td>
</tr>
<tr>
<td>search.cgi</td>
<td>17</td>
</tr>
<tr>
<td>Data Dictionary for Database</td>
<td>19</td>
</tr>
<tr>
<td>Entity Relationship Diagram for Database</td>
<td>22</td>
</tr>
</tbody>
</table>
Executive Summary

The World Wide Web is rapidly changing the way we do things, from how we keep in touch with friends to how we conduct business transactions. Although it has become fairly simple to develop and post an informational web page, the development of web sites that collect data and/or restrict access to data is more complex. The proposed research will involve the investigation of advanced issues related to security, firewalls, encryption, and data integrity and will result in the implementation of a secure web-based data collection application that is protected against hacking and intentional data corruption.

Background

The advanced work in this Honors Project will extend the knowledge and skills that I’ve acquired in the following upper level courses: CPTR 353 Networking, CPTR 342 Web-Based Programming, CPTR 349 Database Systems, and CPTR 448 Advanced Design and Development.
In CPTR 353 Networking we learned about networks and how they work. We also set up a small network in lab using Windows NT and Novell Network. This project will expand on the Networking class by studying firewalls and how to fully secure data transfers between computers and different networks. Networking class never made it far enough to study firewalls and how then protect data from hackers. The database involved will need to be accessed from outside the campus firewall. Therefore to complete the project the firewall must be setup in a way to allow access only to the database and no other part of the schools servers.

CPTR 342 Web-based Programming was the study of how to create HTML documents utilizing Perl and CGI. We created a mock database of CDs with MySQL and then created CGI web pages that allowed users to sort, search, and then “buy” a selected CD. The CD Store project used JAVA to verify user data inputted on HTML forms. This honors project will fully extend this idea to include secure web pages, a larger more robust database, more robust data verification, and user-friendly interfaces. The resulting application will be a real-world application that will be put into actual operation. The idea of security will be emphasized throughout this project as the main area of study.

CPTR 349 Database Management Systems is the study of databases and how to correctly setup and query databases in SQL-99. This project will expand on the knowledge learned in DBMS by creating a relational database using MySQL. The database that will be created must have many constraints and advanced trigger commands programmed into it to make sure the data will not be corrupted.

CPTR 448 Advanced Design and Development teaches how to use and implement different software engineering techniques. We used the ideas taught to specify, design, implement and document a project of large size. The honors project will expand on these ideas
by using the rapid prototyping technique of planning and implementation. This will result in detailed Specifications, Requirements, and Implementation Documents that will aid in the creation and maintenance of the project in question. Along with those documents there will be a User Manual created to allow future updates and maintenance of the database.

Research

The main topic of research will be the network security and firewall setup. Within the study of security the topics of encryption, password authentication, and hack-proof code will be covered. These ideas will then be put to use in the application, the creation of the database with the web front-end.

Application

The Lycoming chapter of Lambda Chi Alpha has been looking for an easy way to maintain information about their alumni and active brothers. So far the only method they have available to them is paper and pencil. This project is designed to give the local Chapter of Lambda Chi Alpha the ability to store alumni and active brother information in an easy to use, secure database. The database will be accessible and updateable from user-friendly web pages on a secure server.

To make this idea possible this Honors Project will utilize the open source Unix Database MySQL currently on Lycoming’s Unix Servers. In conjunction with MySQL the project will implement CGI, Perl, and HTML web pages on a secure server behind Lycoming’s firewall, which will allow users to access and make queries on the data stored in the database. There will be extensive study into creating securely coded web pages and databases. The study on security
will be extended to include firewalls and possible loopholes to close in the firewall so hackers
will not be able to utilize the web pages created to access sensitive data on Lycoming’s servers.

Requirements

The final project will be creating a database in MySQL, web pages to support the
database, and security features to protect the database. The database will contain information
about active brothers of Lambda Chi Alpha and alumni of Lambda Chi Alpha. Some of the
information contained within the database will be IB number, Name, Address, Email address,
Date of Graduation, Office Held, and Phone Number. Some other information might be added at
the request of the current chapter. To protect the data in the database triggers will be setup that
will make sure data input is of correct form and type. There will most likely be two tables, one
for active brothers and one for alumni brothers. The primary key for both these tables will be IB
number because this is unique throughout the fraternity.

The web pages created will be there to make the interface with the database easy and very
interactive. One of the pages created will allow the database administrator to update or modify
information about any brother. Another page will be setup exclusively to do searches on the
database. One such search would be to find brothers that live near you or brothers that graduated
at the same time. To allow the alumni to update the information in the database there will be a
way for them to submit new information to the database administrator.
Bibliography for Honors Project

Celko, Joe. *Joe Celko's SQL for Smarties: Advance SQL Programming*.

DuBois, Paul, Micheal Widenius. *MySQL (OTHER NEW RIDERS)*


Howard, Micheal, David Leblanc. *Writing Secure Code*

Kline, Kevin E., Daniel Kline PhD, Daniel L. Kline. *Sql in a Nutshell: A Desktop Quick Reference*

Krutz, Ronald et al. *The CISSP Prep Guide: Mastering the Ten Domains of Computer Security*

Welling, Luke, Laura Thomson. *PHP and MySQL Web Development*

Smith, Richard E. *Authentication: From Passwords to Public Keys*
The proposed system is being designed to maintain permanent information about active brothers and alumni of Lycoming College's Lambda Chi Alpha fraternity. A web interface will provide restricted access to a secure database through an easy, interactive front-end.

The project will incorporate both MySQL database design and development with Perl, CGI, JavaScript, and HTML web scripting to create an Alumni Database accessible and updateable from the World Wide Web. The first part of the project will be to build the database itself. The database will be created with MySQL, a free open source database currently on Lycoming College’s UNIX server.

There will be four tables of information in the database. The data stored in the brothers table will be the following: Zeta Number (IB), First Name, Middle Name or Initial, Last Name, Surname, Address Line 1, Address Line 2, City, State, Zip Code, Initiation Date, Graduation Date, Email Address, Current Status, and Phone Number. The Primary Key will be the IB or Zeta Number. First Name and Last name will be Not Null while everything else can be set to a NULL value. The brothers table is designed to hold only the current information pertaining to that certain brother. The work table will hold the brother’s work information, such as place of work, title, and the address of where the brother works. The BLDdata table will hold only family tree information which will be the brother’s big’s zeta number and his little’s zeta number. Security passwords and usernames will be held in the username table. All the passwords will be already encrypted in this table.
The database will be accessible from Iota Beta Zeta’s homepage located at
http://www.lycoming.edu/org/izca. Once at the database opening page the user will be asked for a
username and password. If the user hasn’t setup his username and password yet there will be a
link to another page where he can register. A password is required to secure data stored in the
database, to prohibit unauthorized access to addresses and phone numbers of our alumni. After
passing the log-in screen, the user will be able to change his information such as address and
phone number or search for the current information of anybody in the database.

If the user finds his information in the database to be incorrect he will be able to access a
corrections page where he can make corrections to his information and his only. The user will
also be able to remove all of his information except IB, First Name, and Last Name. This new
information will be updated automatically by the CGI script. The search page will be comprised
of a keyword search and a number of areas to browse through. The browse area will have the
following titles: First Name, Last Name, City, State, Initiation Date, Graduation Date, Field of
Work, Work State, and Work City. With these simple browse the user will be able to find
brothers he graduated with, brothers he was initiated with, and brothers who live or work near
him.

There are two other very important pages included in this project. The first is a page to
find lost usernames and passwords. The lost usernames or passwords page will be comprised of
a simple form where the user enters his email address. If the email address is found in the
database the user will be emailed his username and a temporary password that will have to be
updated to gain access to the database again. To change his password there will be a password
c change page. This page will allow the user to change his password anytime. To do this the user
will have to enter his username, old password, and new password. Then the new password and
username will be emailed to the user for his records.

The web pages will all include the latest security measures. These measures include
password encryption, password authentication, and secure script-writing techniques. Everything
will be fully tested to ensure that the web pages and database will pose no threat to the college
community or those people who are included in the database itself.

Modification from Honors Proposal

The honors proposal spoke of firewall modification. This was not needed in the final
project as it was found that the database could be made secure enough with the current
configuration of Lycoming’s firewall. The CGI scripts are accessed the same way any HTML
web page would be accessed; there is no need to make changes to the security already present.

An administration page was to be incorporated into the final project, a page where the
administrator could modify user information and database information. This page was not
completed because of time constraints and because all administration can be done from the
UNIX environment. The user manual will address all administration techniques. The only thing
the administrator will have to do is add new brothers to the database so they can register and
change their own information. I plan on being the primary administrator for at least another
year, in which time I will program an administrator page for the administrator to follow me.
Dataflow Diagram
Title: Alumni.cgi

Purpose: This CGI will display the welcome message and allow the user to login or link to the registration page. If the user tried to register and gives the wrong username or password a link will show up for forgotten passwords or usernames. After user correctly logs in a menu page will appear with three links, Password Change, My Information Page, and Search Page.

Security: This page has security for the password and username. All input from the HTML is checked by a regular expression. In this case the regular expression checks to make sure the input is only alphanumeric, only letters and numbers. The database interface is secure because the password and username for the database is abstracted to another file outside the scope of the HTML directory. Because the password and username for the database are outside the HTML directory only the administrator can read it, nobody can read this file without logging into the LCA UNIX account. The password is encrypted by the MD5 algorithm before being crosschecked with the encrypted password in the username table of the database.

Screenshots: Main Page

The Iota-Beta Zeta Alumni Database is an online meeting place for brothers of Lambda Chi Alpha at Lycoming College to stay in contact with each other. We all know how hard it can be to stay in touch after college, this database is intended to help with that situation. Stored within our database is the latest contact information for all initiated brothers of Lambda Chi Alpha. We hope you enjoy this service and don't misuse it.
**Title: forgot.cgi**

**Purpose:** This page is designed to allow the user to find his forgotten username or password. The user enters his email address into the form and hits submit. If he has registered with that email address he will be emailed with his username and a new password to login to the database. After he enters the site again he will be prompted to change his password.

**Security:** The security on this page consists of a full check of the email address entered. Email addresses go through a thorough check for form, content, and special security characters.

**Screenshot:**

Iota-Beta Zeta  
Alumni Database  
Username/Password Lookup  

If you forgot your Username or Password  
Fill in your Email Address here and it will be emailed to you.  
You will have to change your password because it will be changed!

Submit  Reset

Iota-Beta Zeta  
Alumni Database  
Username/Password Lookup  

An email has been sent to you with the requested information  

Main Alumni Page
Title: pass.cgi

Purpose: Pass.cgi is designed to allow the user to change his password. The user enters his username, old password, and new password twice. His password is then changed for the next time he logs in to the database.

Security: This page does pattern matching on all input. The username and passwords must be all alphanumeric. The email address goes through a check for form, special characters, and security checks. This security check on the email address is needed because the email function in the CGI script interacts with the UNIX environment. If the user enters an email address with UNIX commands embedded in it, the user could email himself sensitive data. The email address check removes all security leaks that could occur from somebody trying to do this.

Screenshots:

Iota-Beta Zeta Change Password

Username:
Old Password:
New Password:
Retype New:

Submit | Reset

Iota-Beta Zeta Change Password

Your password has been updated.

You will have to sign in again to access any other pages.

Main Alumni Page
Title: myinfo.cgi

Purpose: My Info page is set up to allow the user the ability to update his information at anytime. The user does this by changing or adding information to the form. After he is done he will submit the changes to the CGI which will update all tables in the database.

Security: This page has full security on all fields, both with JavaScript and with pattern matching. No illegal dangerous characters are allowed such as the """" or the quote. Each field has its own check; some fields must be all characters while others must have specific formats.
Title: regist.cgi

**Purpose:** Regit is short for register. This page allows alumni and active brothers to register for a username to access the database. To do this they must supply their first name, last name, email address, and password twice. After the user is identified by the database he is added to the username table with his password encrypted and then emailed his username and password. The user must already be in the brothers table of the database, this ensures that only initiated brothers of Iota-Beta Zeta are allowed to register for access to the database.

**Security:** This page incorporates pattern matching on all fields. It also has JavaScript to assure that all fields are filled in and the user agrees to the terms and conditions of the database. The password is encrypted with MD5 before being saved in the username table. The email address is fully checked by a security algorithm.

**Screenshot:**

*Welcome to the Iota-Beta Zeta Alumni Registration Page*

You will be able to sign up for your username and password to access the Iota Beta Zeta Alumni directory and listings.

1. The data stored on the Iota-Beta Zeta Alumni page is to be used and viewed only by Alumni of Iota Beta Zeta.
2. Nobody will be allowed to modify or change your information without your consent agreement.
3. Your information will be kept private for the use of Iota-Beta Zeta and its Alumni only.
4. Your information will not be sold by Iota-Beta Zeta to anybody.

Check this box if you agree to the terms above □

- First Name
- Last Name
- Username
- Email Address
- Password
- Retype Password

[Submit] [Reset]
Title: search.cgi

Purpose: Search.cgi allows the user to search specific keywords or browse by categories to find their old friends and brothers. The keyword search is limited to either first name, last name, IB, state, or city. And the browse categories are first name, last name, city, state, initiation date, graduation date, field of work, work city, and work state. After browsing or searching to a list of brothers, those brothers on the screen can be sorted by IB, first name, or last name. Once the user finds somebody they want to know more about, they can click on that person’s IB to be shown that person’s information. And by clicking on that person’s email address they can send an email to that person.

Security: This page doesn’t require that much security. The password and username are checked from the temporary cookie stored in memory against the username table. This cookie will be deleted immediately when the user exits the browser. The keyword searches can only be alphanumeric so no illegal searches can occur.

Screenshots:

Iota-Beta Zeta

Search

Return to Main Alumni Page | Go to My Info Page | Main Search Page
<table>
<thead>
<tr>
<th>Name of Data Element</th>
<th>Description</th>
<th>Narrative</th>
</tr>
</thead>
<tbody>
<tr>
<td>ica_data</td>
<td>Database containing four tables: brothers, work, BLData, username</td>
<td>The whole Database containing all information about Iota-Beta Zeta Alumni and Active Brothers</td>
</tr>
<tr>
<td>brothers</td>
<td>Records: ib, first_name, last_name, middle_name, surname, street1, street2, city, state, zip, grad, init_date, status, phone, email</td>
<td>Table in Database containing contact information about brother, also includes graduation date, initiation date, and status</td>
</tr>
<tr>
<td>work</td>
<td>Records: ib, title, company, work_phone, work_address, work_city, work_state, work_zip, fieldofwork</td>
<td>Table in Database containing current work information about brother.</td>
</tr>
<tr>
<td>BLdata</td>
<td>Records: ib, big, little</td>
<td>Table containing family tree information for brothers in Iota-Beta Zeta</td>
</tr>
<tr>
<td>username</td>
<td>Records: ib, username, password</td>
<td>Table containing security information for brothers registered to use database.</td>
</tr>
<tr>
<td>Name of Data Element in brothers table</td>
<td>Description</td>
<td>Narrative</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ib</td>
<td>10-digit unsigned integer</td>
<td>Unique number given to every initiated brother of Iota-Beta Zeta. This is the Primary Key for the Entire Database</td>
</tr>
<tr>
<td>first_name</td>
<td>20 character string</td>
<td>The First name of a brother, cannot be NULL.</td>
</tr>
<tr>
<td>middle_name</td>
<td>20 character string</td>
<td>The Middle name if specified of a brother.</td>
</tr>
<tr>
<td>last_name</td>
<td>20 character string</td>
<td>The Last name of a brother, cannot be NULL</td>
</tr>
<tr>
<td>suffix</td>
<td>5 character string</td>
<td>Used to store titles like Dr. or surnames like Jr.</td>
</tr>
<tr>
<td>street1</td>
<td>50 character string</td>
<td>First line of address</td>
</tr>
<tr>
<td>street2</td>
<td>50 character string</td>
<td>Second line of address if needed</td>
</tr>
<tr>
<td>city</td>
<td>50 character string</td>
<td>City or City and Country of address</td>
</tr>
<tr>
<td>state</td>
<td>2 character string</td>
<td>The State of the address, or nothing if not in USA</td>
</tr>
<tr>
<td>zip</td>
<td>10 character string</td>
<td>Postal Zip Code</td>
</tr>
<tr>
<td>init_data</td>
<td>Date</td>
<td>The initiated date of brother stored as YYYY-MM-DD</td>
</tr>
<tr>
<td>grad</td>
<td>10-digit unsigned integer</td>
<td>Graduation Date of brother stored as YYYY-MM-DD</td>
</tr>
<tr>
<td>phone</td>
<td>20 character string</td>
<td>Current or last known phone number of brother</td>
</tr>
<tr>
<td>email</td>
<td>80 character string</td>
<td>Email address of brother</td>
</tr>
<tr>
<td>status</td>
<td>30 character string</td>
<td>Current status of brother: Active, Alumni, Missing or Deceased</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name of Data Element in work table</th>
<th>Description</th>
<th>Narrative</th>
</tr>
</thead>
<tbody>
<tr>
<td>ib</td>
<td>10-digit unsigned integer</td>
<td>Unique number given to every initiated brother of Iota-Beta Zeta. This is the Primary Key for the Entire Database</td>
</tr>
<tr>
<td>title</td>
<td>20 character string</td>
<td>Brother’s title at his work place</td>
</tr>
<tr>
<td>work_phone</td>
<td>20 character string</td>
<td>Brother’s work place phone number</td>
</tr>
<tr>
<td>fieldofwork</td>
<td>50 character string</td>
<td>Brother’s Field of Work</td>
</tr>
<tr>
<td>work_address</td>
<td>70 character string</td>
<td>Address of Work Place</td>
</tr>
<tr>
<td>work_city</td>
<td>50 character string</td>
<td>City of work place</td>
</tr>
<tr>
<td>work_state</td>
<td>2 character string</td>
<td>State or work place</td>
</tr>
<tr>
<td>work_zip</td>
<td>10 character string</td>
<td>Work place postal code</td>
</tr>
<tr>
<td>company</td>
<td>20 character string</td>
<td>Company name of work place</td>
</tr>
<tr>
<td>Name of Data Element in BLdata table</td>
<td>Description</td>
<td>Narrative</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ib</td>
<td>10-digit unsigned integer</td>
<td>Unique number given to every initiated brother of Iota-Beta Zeta. This is the Primary Key for the Entire Database</td>
</tr>
<tr>
<td>big</td>
<td>10-digit unsigned integer</td>
<td>lb of fraternal big brother</td>
</tr>
<tr>
<td>little</td>
<td>10-digit unsigned integer</td>
<td>lb of fraternal little</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name of Data Element in username table</th>
<th>Description</th>
<th>Narrative</th>
</tr>
</thead>
<tbody>
<tr>
<td>ib</td>
<td>10-digit unsigned integer</td>
<td>Unique number given to every initiated brother of Iota-Beta Zeta. This is the Primary Key for the Entire Database</td>
</tr>
<tr>
<td>username</td>
<td>20 character string</td>
<td>Username of registered user</td>
</tr>
<tr>
<td>password</td>
<td>35 character string</td>
<td>MD5 128 bit encrypted password of user, stored as string of Hexadecimal code</td>
</tr>
</tbody>
</table>
# User Manual for maintenance of the Iota-Beta Zeta Alumni Database

## Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Logging into the Unix System</td>
<td>1</td>
</tr>
<tr>
<td>2. Working With X-Win32</td>
<td>2</td>
</tr>
<tr>
<td>3. Working With UNIX</td>
<td>3</td>
</tr>
<tr>
<td>4. Accessing MySQL</td>
<td>5</td>
</tr>
<tr>
<td>5. Getting Information From and Putting Information into the Database</td>
<td>6</td>
</tr>
<tr>
<td>6. Updating information in the Database</td>
<td>7</td>
</tr>
<tr>
<td>7. Restoring and Backing up the Database</td>
<td>8</td>
</tr>
</tbody>
</table>
Ch. 1 Logging in to the UNIX System

The alumni database is completely set-up in the UNIX system of Lycoming College. To do any maintenance on the alumni database you must first be logged into the system, for example through X-Win32 or Telnet. First we will work through using X-Win32, the current system interface found in all of Lycoming College’s computer labs. Then we will work on gaining access to the UNIX account through Telnet, a program found in all Windows operating systems.

To start X-Win32 go to the Start menu on a computer lab computer. Go to Course Programs, Programming, and then X-Win32. A screen will open up with the X-Win login screen. Enter our user account name here, LCA. The next screen will ask for the password. Enter the password for the LCA account. If you do not know the password you can get it from the Unix Administrator in the office of Communication Technology. Now you’re logged in to the UNIX system “lyco.lycoming.edu”. The next chapter will explain how to work in X-Win32.

Some maintenance operations of the database and CGI scripts that interact with the database cannot be done from the X-Win32 interface. To do these operations you must be using a telnet-type program. The telnet program that comes with all Microsoft Windows operating systems is easy to use. To access the LYCO system, go to the Start Menu and then Run. The Run command prompt will pop-up. Enter this command here exactly, telnet lyco.lycoming.edu. This command will open a telnet window to the LYCO server of Lycoming College, such as the one in Figure 1. Now enter the account name, LCA. It will ask for the password. Enter the LCA account password to see the welcome screen. Under the welcome text is some information of interest, the last unsuccessful login and the last login for the account. You can use this information to see if anybody was trying to hack into the account.
CH. 2 Working with X-Win32

Now that you know how to login to X-Win32 you should begin familiarizing yourself with its operations. At the bottom of the X-Windows interface you should see what looks like Figure 2. There are two icons of major interest, the first being the third icon from the left. This icon looks like a file cabinet, and opens the X-Win file browser. In this browser you will be able to rename, change permissions, move, and open files in your UNIX account. The next important icon is hidden in a menu above the fourth icon from the left, what looks like a paper and pencil. This menu hides the Terminal icon which opens a Terminal window; a Terminal window is just like a Telnet connection.

The file browser will look like Figure 3. In this case we are in the lcSQL directory; this directory contains all the files that are associated with MySQL and its set-up. The file browser will allow you to easily change file permissions, open files to be edited, and copy files to different directories. To open a file for editing simply right click on its icon and select OPEN from the options. If you double-click on the icon it will try to run the file; this will not work because the “.sql” files need to be run at the command prompt and the “.cgi” files need to be run from a web browser.
CH. 3 Working with UNIX

The file browser will only allow you to do simple operations. To do more complex operations you're going to have to use the terminal window and UNIX commands. We'll work through a couple of very useful commands that you will have to know to work in UNIX. The first is *ls*; *ls* is the same as *dir* in DOS. This command will give you a simple listing of files and directories in the current directory. As with all commands in UNIX you can add options onto the end of the command to make it do more. A useful combination is *ls -a*; this will give you all files and directories, including hidden, with file permissions in front of the file name. File permissions will be explained a little bit later.

Now that you know what directories and files are located in your current directory, you may want to change directories. To do this you must use the *cd* command, which stands for change directory. The change directory command, *cd*, is combined with a directory name to move into that directory. An example is *cd public_html* or more complex *cd public_html/cgi-bin/alumni*. The first command will move down one directory into the public_html directory, while the second command moves you all the way down to the alumni directory in the cgi-bin. This is where you will find most of the scripts for the web pages of the database. To move up a directory to the directory you just came from you add two periods to the *cd* command, which will look like this "*cd ..". The *ls* and *cd* commands will look like Figure 4 in UNIX.
<table>
<thead>
<tr>
<th>File</th>
<th>Mode</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Owner</th>
<th>Group</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcaSQL</td>
<td>74K</td>
<td></td>
<td>22 Apr</td>
<td>17:10</td>
<td></td>
<td></td>
<td>r-x---------</td>
</tr>
<tr>
<td>public_html</td>
<td>10K</td>
<td></td>
<td>25 Jan</td>
<td>11:20</td>
<td></td>
<td></td>
<td>r--rw-r-----</td>
</tr>
<tr>
<td>sample.db</td>
<td>30K</td>
<td></td>
<td>15 Feb</td>
<td>10:30</td>
<td></td>
<td></td>
<td>r----------x-</td>
</tr>
</tbody>
</table>

Now that you can navigate through the UNIX file system and also know what is in the directory your in, you should be able to move the lcaSQL directory off the main LCA directory. The contents should be the same as the last line in Figure 4. To open a file for editing in UNIX you can use the pico editor. To open files with pico simply add the file name to the end of pico.

So, to open the create_work.sql file, type pico create_work.sql and the file will open in the pico editor. There are commands at the bottom of the pico screen; you will have to familiarize yourself with these commands to do anything in pico. To see the pico screen look at Figure 5 on the next page. The "C" in front of each command stands for "ctrl", so to save the file you would hold down the ctrl key and then hit O.

Then to exit you would hold down the ctrl key and hit X. You should be able to get the basic idea very quickly.
If you need to know more UNIX commands than the ones described here, go to the web. There are many web sites set-up that do nothing more than list UNIX commands and what they do. We will now move on to working with MySQL in UNIX.

**CH. 4 Accessing MySQL**

To access MySQL you must login through the UNIX system. To login to MySQL type `mysql -u lca -p lca_data` at the command prompt in UNIX. This will show the screen in Figure 6. The `-u` means you’re supplying the user name, in this case `lca`. And the `-p` means you’re using a specific database, in this case `lca_data`. Always use this means of logging into MySQL, that way you will always access the correct database. After you hit enter to send the command, it will ask you for the password. Simply type the password in to gain access to MySQL.
CH. 5 Getting Information From and Putting Information into the database

After logging in you can modify or query any of the tables in the lca.data database. We will discuss making tables and backing up data in the next chapter. To query a table you will have to use the SELECT statement. A simple SELECT statement to find all brothers with last names starting in “S” looks like this:

```
SELECT first_name, last_name
FROM brothers
WHERE last_name like "S%";
```

Try typing this command at the MySQL prompt. You should see a list of all the brothers with last names starting with “S”. There are three basic parts to a SELECT statement, the first line asks for what fields you want to retrieve from the table. The second line specifies what table or tables you want this data from. And the third line limits the select statement to specific values, in this case any last name that starts with “S” with anything after it. The “%” means anything, so if you typed “Spe%”, the returned results would be all brothers with last names starting with “Spe”. Capitalization does not matter in any SQL statement. To find a specific value, replace the “like” with an equal sign, “=”. To find out about more complex SELECT statements I suggest either searching the web or going to the library and reading the book titled MySQL by Paul DuBois.

Another useful command in MySQL is DESCRIBE. We use DESCRIBE to describe what tables are comprised of. If we enter DESCRIBE brothers at the command prompt it will show a description of the brothers table. This is useful information to know when we want to add another brother to the table; we will need the exact field names. To add a brother to the brothers table you will have to use the following command:

```
INSERT INTO brothers (ib, first_name, last_name)
VALUES (773, "Joshua", "Speicher");
```

This command will insert the IB 773, first name “Joshua”, and last name “Speicher” into the brothers table. You will have to do this for all new initiated brothers. You must remember that
the IB must be unique; if not, the INSERT statement will give you an error. To add other information like address or phone number, simply add the field names after last_name and add the values after the last name. Always separate values and field names with commas, and add quotes around values that are varchar, or alphanumeric.

CH. 6 Updating information in the Database

To update a brother’s information, say if a brother changes his email address, you will have to use the UPDATE statement. To update the email address for a brother use this command:

```
UPDATE brothers
SET email = "name@dot.com"
WHERE ib = 773;
```

This statement will change the brother with IB 773 email address to “name@dot.com.” You can also modify many brothers information at once. Say you want to change the status of recently graduated seniors; you can change the above statement to look like this:

```
UPDATE brothers
SET status = "alumni"
WHERE grad = "2002";
```

This will update every brother who graduates in 2002 to alumni status.

CH. 7 Restoring and Backing up the Database

Making backups of the database is a must, and should be done on a weekly to biweekly basis. To do such a backup you must be logged into UNIX from Telnet. Go to the lcaSQL directory and then the BackUp directory in the lcaSQL directory. To do this you can type cd lcaSQL/BackUp right after you log in. To make a backup of the database and its table design, you will have to use the mysqldump command from the UNIX prompt. Type mysqldump --add-drop-table --add-locks lca_data > lcaDataBackUp at the command prompt to make a full
backup of the database and all information contained within. The file created will look similar to, but not the same as, Figure 7.

**Figure 7**

```sql
```

<table>
<thead>
<tr>
<th>Table: 'Ica_data'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>age</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table: 'brothers'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>age</td>
</tr>
</tbody>
</table>
```

The file created with mysqldump will allow you to restore the database if it would ever be deleted. Now to restore the backup of the database you will have to use the following command, `mysql -u lca -p lca_data < lcaDataBackUp`. It will ask for the password, type the password in to complete the restore. The database will be fully restored to the exact same condition it was in at the last backup. This is why it is so important to backup the database on a regular schedule.
# Appendix A

## Table of Contents

For Appendix A

<table>
<thead>
<tr>
<th>File</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>alumni.cgi</td>
<td>1</td>
</tr>
<tr>
<td>pass.cgi</td>
<td>8</td>
</tr>
<tr>
<td>myinfo.cgi</td>
<td>15</td>
</tr>
<tr>
<td>regit.cgi</td>
<td>29</td>
</tr>
<tr>
<td>search.cgi</td>
<td>40</td>
</tr>
<tr>
<td>formLib.js</td>
<td>63</td>
</tr>
</tbody>
</table>
use DBI; # For accessing Database
use strict; # Keeps variables local or global
use CGI; # Needed for CGI.pm
use MD5; # Needed for 128bit encryption

my $md5 = new MD5;
my $cgi = new CGI;

my($User) = $cgi->cookie("user_name") =~ /\w+/; # Gets username from Cookie
my($Pass) = $cgi->cookie("password") =~ /\w+/; # Gets Password from Cookie
my($UserName) = $cgi->param("user_name") =~ /\w+/; # Gets Username from environment/form
my($Password) = $cgi->param("password") =~ /\w+/; # Gets Password from enviroment/form
my($Return) = $cgi->param("return"); # If returning to alumni menu?
my($db_return);
my($error) = 0; # If not logged in

# START OF DBI SETUP FOR USERNAME AND CONNECTION PARAMS
my($sth);

my($host_name, $user_name, $password) = (undef, undef, undef);
my($db_name) = "lca_data";
my($dbh, $dsn);
my($attr) = (RaiseError => 1);

$dsn = "dbi:mysql:$db_name";
if(!$UserName || !$Password) & ( !$Return ){
    print $cgi->header(-type => "text/html");
    print $cgi->start_html(-title=>"Iota Beta Zeta Alumni Directory");
    StartPage();
    PrintTable();
}
else{
    if($Password eq "IotaBeta") || ($Password eq "iotabeta"){
        #If user forgot his password he must change it
        print $cgi->header(-type => "text/html");
        print $cgi->start_html(-title=>"Iota Beta Zeta Alumni Directory");
        StartPage();
        print "<center>";
        print $cgi->h3("You must update your password before you can continue.");
        print "<a href="http://lyco.lycoming.edu/lca/cgi-bin/alumni/pass.cgi">Update Password</a>";
        print "</center>";
    }
    else{
        #User doesn't have to change password
        if(!$return){
            #If user is returning from another page and is already logged in
            my $UserCookie = $cgi->cookie(-name => "user_name",
                                      -value => $UserName);   #Set's username cookie
            EncryptPass();
            my $PassCookie = $cgi->cookie(-name => "password",
                                           -value => $Password);   #Set's password cookie
            print $cgi->header(-type => "text/html", -cookie => [$UserCookie, $PassCookie] );
        }
        else{
            #Not returning from another page
            print $cgi->header(-type => "text/html");
        }
        print $cgi->start_html(-title=>"Iota Beta Zeta Alumni Directory");
        StartPage();
        #Start the page with a title and the table
    }
CheckDB();                      #Check for username and password
if($db_return->{ib}){
    PrintMenu();                      #Print the table of options
}
else{
    $error = 1;                      #else not logged in right
    PrintTable();                   #Print table with error message
}
}

# Encrypts the password for crosschecking with the username database
sub EncryptPass{
    # this encrypts the password #
    $PassWord =~ tr/A-Z/a-z/;        
    $md5->add($PassWord);            
    $PassWord = $md5->hexdigest();   
}

#Purpose: Checks the Username database for the username and encrypted password
#Pre: Database must be built with username table
#Post: user's IB is saved in variable "$ib"
sub CheckDB{

    $dbh = DBI->connect ($dsn, $user_name, $password, {attr});

    # Checks for ib in username table, can't have two of the same usernames
    $sth = $dbh->prepare(qq{
        SELECT ib FROM username
        WHERE user_name = ? AND password = ?
    });

    $sth->execute($UserName, $PassWord);

    $db_return = $sth->fetchrow_hashref();
    #if password and username are not in cookie
    if(!$db_return->{ib}){
        $sth->execute($User, $Pass);
        $db_return = $sth->fetchrow_hashref();
    }
$dbh->disconnect();

#Purpose: Starts the HTML and prints the title for the page. Also has the JavaScript embedded in it.
#Pre: None
#Post: Page is started with javascript and header
sub StartPage{

    print <<ENDHTML;
<script src="http://lyco.lycoming.edu/~lca/js-lib/FormLib.js"></script>
<script>
    function validateForm(form){
        requiredText = new Array( "user_name", "password");
        return requiredValues(form, requiredText)
        && checkProblems();
    }
    //---
</script>

<p align="center"><font size="6">Iota-Beta Zeta</font>
Alumni Database</font></p>
ENDHTML
};

#Purpose: Prints the Menu of options after logging in with hidden fields for passing username and password
#Pre: None
#Post: menu of options is displayed on the screen
sub PrintMenu{

    print <<ENDHTML;
    <div align="center">
    <center>
    <table border="1" cellpadding="5" style="border-collapse: collapse" bordercolor="#cccccc" width="100%"
    id="AutoNumber1">
    <tr>
        <td width="50%" bgcolor="#FFFF00">
        <p align="left" color="#6600CC">To see and modify your user information go to your</p>
        <a href="http://lyco.lycoming.edu/~lca/cgi-bin/alumni/myinfo.cgi">information page</a>.</font></td>
    ENDHTML
}
To change your password go to http://lyco.lycoming.edu/~lca/cgi-bin/alumni/pass.cgi

To search the Iota-Beta Zeta Alumni Database go to the Search Page http://lyco.lycoming.edu/~lca/cgi-bin/alumni/search.cgi

Purpose: Prints the Table for the Welcome page, gives an error message if username or password are incorrect.
Pre: HTML must be started already
Post: Print’s html for main alumni page

sub PrintTable{
print <<ENDHTML:
<table border="1" cellspacing="0" width="100%" id="AutoNumber1" height="354" bgcolor="#FFFFFF"
style="border-collapse: collapse" bordercolor="#111111" cellpadding="2">
<tr>
<td width="11%" height="207"><form method="POST" action="alumni.cgi" onSubmit="return validateForm( this );">
</form></td>
</tr>
</table>
ENDHTML

if($Error){
    #If not logged in correctly

print <<ENDHTML
<center>
<font size="4">Your Username or Password was not found.</font><br>
Username: <input type="text" name="user_name" size="20"><br>
Password: <input type="password" name="password" size="20"><br>
<input type="submit" value="Submit" name="B1">
</center>

ENDHTML

}
#!/usr/bin/perl
#note: there should be a "-T" at end of first line, but getting WEAK error so not using like 1 should be

# Title : Forgot Username or Password Page
# Author: Joshua Speicher
# Last Update: 3/25/02
# Purpose: This page is a simple password or username forgot page, it will just email
# the user his username and reset his password to a default value
# Pre : None
# Post : If username and password found givin menu options and allowed to continue on

use DBI;
use CGI;
use strict;
use MD5;

my($md5) = new MD5;
my($cgi) = new CGI;

my($email) = validate_email_address( $cgi->param("email")); #Validated email address
my($username);
my($password) = "IotaBeta";
my($id);
my($firstName);
my($lastName);
my($db_return1);
my($db_return2);

#START OF DBI SETUP FOR USERNAME AND CONNECTION PARAMS
my($sth);

my($host_name, $user_name, $password) = (undef, undef, undef);
my($db_name) = "lca_data";
my($dbh , $dsn);
my($attr) = (RaiseError => 1);

$dsn = "dbi:mysql:$db_name";
### DBI SETUP

```perl
$dsn = ":host_name=$host_name" if $host_name;
$dsn = ":mysql_read_default_file=/student/stu/1ca/alumni.cnf";
```

```perl
#END OF DBI SETUP

print $cgi->header();

###this is the Mail Section of the program
if(!$Email){
    #If no email address given
    StartPage(); #start the html
    printForm(); #Print the form
}
else{
    CheckDB(); #check db for email address and change password
}
print $cgi->end_html();
```

### Validate Email Address

**Purpose:** Checks email address for correct format

**Pre:** none

**Post:** Email address is cleared of any illegal characters or formatting

```perl
sub validate_email_address{
    my $addr_to_check = shift;
    $addr_to_check =~ s/([^@:][^\s]+@.\d+\.[^\s]+)\s*\[^[^\t]*\t]*[\t]*\$/1/g;

    my $esc = \"\\\";
    my $space = \"\040\";
    my $ctrl = \"\000-\037\";
    my $dot = \"\.";
    my $nonASCII = \"\x80-\xff\";
    my $CRlist = \"\013\015\";
    my $letter = \"a-zA-Z\";
    my $digit = \"\d\";
    my $atom_char = qq{ [^$space<>@,:;:][\\\$esc$ctrl$nonASCII] };
    my $atom = qq{ $atom_char+ };
    my $byte = qq{ [?:1?$digit?$digit |
                2[0-4]$digit |
                25[0-5] ] };
    my $qtext = qq{ [$esc$nonASCII$CRlist] };
```
my $quoted_pair = qq( \$esc {"$nonASCII" } );
my $quoted_str = qq( " (?: $text | $quoted_pair )* " );

my $word = qq( (?: $atom | $quoted_str ) );
my $ip_address = qq( \$byte (?: $dot $byte ) {3} \$\ );
my $sub_domain = qq( [\$letter\$digit]
    [\$letter\$digit-][0-6]\$\$letter\$digit] );
my $top_level = qq( (?: $atom_char ){2,4} );
my $domain_name = qq( (?: $sub_domain $dot )+ $top_level );
my $domain = qq( (?: $domain_name | $ip_address ) );
my $local_part = qq( $word (?: $dot $word )* );
my $address = qq( $local_part \$ $domain );

return $addr_to_check =~ /"$address$/i ? $addr_to_check : "";

## Encrypt the password for crosschecking with the username database
sub EncryptPass{
    #################################################################
    # this encrypts the password
    #################################################################
    $PassWord =~ tr/A-Z/a-z/;
    $md5=$/add($/PassWord);
    $PassWord=$/md5/=hexdigest();
    #################################################################

    #Purpose: Emails user desired information, username, new password, and now to change it
    #Pre: $Email, $FirstName, $LastName, $PassWord must be defined
    #Post: User is emailed his username and new password
    sub Mail{
        open MAIL, "| /usr/lib/sendmail -t -i";
        print MAIL <<END_MAIL;
        TO: $Email
        FROM: IotaBetaZeta\Yahoo.com
        REPLY-TO: IotaBetaZeta\yahoo.com
        SUBJECT: Iota Beta Zeta Forgot Password or Username

        $FirstName $LastName,
        Here is your Username and Password you will have to update your password before being able to enter the web site again.

END_MAIL;
    }
}
To do that visit http://lyco.lycoming.edu/-lca/cgi-bin/alumni/pass.cgi
and update your password.

Your Username is: $UserName
Your Temporary Password is: $PassWord

Thank You,
From all the Brothers of Iota-Beta Zeta

P.S. If you have any questions or recieved this email by error send comments to IotaBetaZeta\@Yahoo.com

END_MAIL
close MAIL
}

#Purpose: Checks the Username database for the username and encrypted password
#Pre: Username and Password must be defined
#Post: $ib is saved in global variable

sub CheckDB{
    $dbh = DBI->connect ($dsn, $user_name, $password, \%attr);

    # Checks for ib in username table, can't have two of the same usernames
    $sth = $dbh->prepare(qq{
        SELECT first_name, last_name, ib FROM brothers
        WHERE email = ?
    });
    $sth->execute($Email); #Execute querey with email address entered

    $db_return1 = $sth->fetchrow_hashref(); #Get hash of data from brothers table

    $FirstName = $db_return1->{first_name}; #set $FirstName to first_name got from table
    $LastName = $db_return1->{last_name}; #set $LastName to last_name got from table
    $ib = $db_return1->{ib}; #set $ib to ib got from table

    if($ib){
        #If ib found in database
        $sth = $dbh->prepare(qq{
            SELECT * FROM username
            WHERE ib = ?
        });
        $sth->execute($ib);
    }
if($db_return2->{user_name}) {
    // If user_name was found in username
    $UserName = $db_return2->{user_name};
    // Set $UserName equal to username from table
    $Mail();
    $sth = $dbh->prepare('UPDATE username
        SET password = ?
        WHERE ib = ?
    ');
    $PassWord = EncryptPass();
    // Encrypt the Password
    $sth->execute($PassWord, $bit);
    // Update the username table with new information
    $PrintThank();
}
else {
    // If IB not found
    print '<Center>
        That Email Address was not found.
    </Center>';
    print $cgi->h2("You may not have registered yet.");
    print "<a href="/http://lyco.lycoming.edu/-lca/cgi-bin/alumni/regit.cgi"">
        Registration Page</a>.</font>
    print '</center>
    if no IB found in username, hence not registered
}
else {
    // That email address was not found in our records
    print '<Center>
        That email address was not found in our records
    </Center>
    print "<a href="/http://lyco.lycoming.edu/-lca/cgi-bin/alumni/regit.cgi"">
        Registration Page</a"></font>
    print '</center>
    $dbh->disconnect();
}
#Purpose: Starts the HTML and prints the title for the page. Also has the JavaScript Embedded in it.
#Pre: none
#Post: none
sub StartPage{
    print "<html"
Iota-Beta Zeta Username or Password Lookup

function validateForm(form) {
  requiredText = new Array( "email");
  return requiredValues(form, requiredText)
    && checkProblems();
}

//--->

Iota-Beta Zeta Alumni Database Username/Password Lookup

ENDHTML

#Purpose: This Prints the Form for Email Address Lookup
#Pre: HTML must be started already
#Post: None
sub PrintForm{
  print "<FORM action=" on Submit="return validateForm( this );">
  <CENTER>
  If you forgot your Username or Password<br>
  Fill in your Email Address here and it will be emailed to you.<br>
  You will have to change your password because it will be changed!</p></CENTER>
  <p align="center"> <input type="text" name="email" size="15"></p>
  <p align="center"> <input type="submit" value="Submit" name="B1">
    <input type="reset" value="Reset" name="B2"></p>
</FORM>
<p align="center">&nbsp;</p>
END
}
#Purpose: To print the thankyou information
#Pre: HTML must be started
#Post: NONE
sub PrintThank{  
  StartPage();
  print "<Center>";
  print "<h2>"."An email has been sent to you with the requested information"");
  print "<a href="http://lyco.lycoming.edu/~lca/cgi-bin/alumni/alumni.cgi">Main Alumni Page</a>";
  print "</Center>";
}
use DBI;
use CGI;
use strict;
use MD5;

my($md5) = new MD5; #Used for encrypting
my($cgi) = new CGI; #Used for using CGI script

my($UserName) = $cgi->cookie("user_name") =~ /\([\w.]+)/; #Gets username from Cookie
my($Password) = $cgi->cookie("password") =~ /\([\w.]+)/; #Gets Password from Cookie

#the next two arrays for storing field information, very messy looking!
my @Fields = ("first_name", "last_name", "middle_name", "suffix", "phone", "email", "street1", "street2", "city", "state", "zip", "init_date", "grad");

my @WorkFields = ("title", "work_phone", "fieldofwork", "work_address", "work_city", "work_state", "work_zip", "company");

my($Big) = $cgi->param("big") =~ /\([\d]+\)/; #Match only Numbers
my($Little) = $cgi->param("little") =~ /\([\d]+\)/; #Match Only Numbers

my $data; #HASH of user information, easy access and storage!!
my $work; #HASH of work data, again easy to use and update, not too messy
my $field; #used with for loops
# The next two foreach loops get the data from the form
foreach $field (@fields) {
    if($cgi->param($field) =~ /([-\w\s.\(\)\-\@\]+)) {  # Matches only
        $data{$field} = $1;  # if correct set to data
    } else {
        $data{$field} = "";  # if not correct make sure empty
    }
}  #end foreach $field (@fields)

foreach $field (@workFields) {
    if($cgi->param($field) =~ /([-\w\s.\(\)\-\@\]+)) {  # Matches only char, spaces, . ( ) - @ ,
        $work{$field} = $1;  # if in correct format with pattern matching then set
    } else {
        $work{$field} = "";  # If not in correct format make sure it's empty
    }
}  #end foreach $field (@workFields)

my($ib);  # the users IB stored in one easy to access variable
my($db_return1);  # Used to fetch data from MySQL
my($db_return2);  # Used to fetch more data from MySQL
my($db_return3);  # Used to fetch so much more data from MySQL

# START OF DBI SETUP FOR USERNAME AND CONNECTION PARAMS
my($sth);

my($host_name, $user_name, $password) = (undef, undef, undef);
my($db_name) = "lca_data";

my($dbh, $dsn);
my($attr) = (RaiseError => 1);

$dsn = "dbi:mysql:$db_name";
$dsn .= ":\host_name=$host_name" if $host_name;
$dsn .= ":mysql_read_default_file=/student/stu/lca/alumni.cnf";

# END OF DBI SETUP

# Print Header
print $cgi->header(-type => "text/html");
print $cgi->start_html(-title=>"Your Information Page");
CheckDB();  # Checks the Databases for user FB and Password
StartPage();  # Starts the page with title and greeting
if($fB){  # If FB found do this stuff
    if( !$_cgi->param("submit") ){
        GetData();  # If user visiting for first time
    }
    else{
        CheckData();  # Checks data in perl just incase java is turned off
        PutData();  # If user wants to update info then go ahead and do so
        GetData();
    }
    PrintForm();  # Print the pretty Form!
}
else{  # Not a user or cookies not enabled
    print $_cgi->center;
    print $_cgi->h3("Either you are not Signed in or you do not have cookies enabled.");
    print $_cgi->h3("Please return to the main alumni page and sign in.");
}

print "<center>");
print "<a href="http://lyco.lycoming.edu/~lca/cgi-bin/alumni/alumni.cgi?return=true">Return to Main Alumni Page</a>");
print "| ");
print "<a href="http://lyco.lycoming.edu/~lca/cgi-bin/alumni/search.cgi">Go to the Search Page</a>");
print "</center>");
print $_cgi->end_html();  # Finish it all off with some html brackets

# Purpose: To check input for special characters and form
# Pre: all variables must already be read in
# Post: if error, then error message. If all good then nothing
sub CheckData{
}

# Purpose: Checks email address for correct format
# Pre: none
# Post: Checked email address without any "dangerous" content
sub validate_email_address{
my $addr_to_check = shift;
$addr_to_check =~ s/\([^\^t\^t-\^t=\^t]*\)/\1\2/si/g;

my $esc        = '\\\';
my $space      = '\040';
my $ctrl       = '\000-\037';
my $dot        = '\.';
my $nonASCII   = '\x80-\xff';
my $CRlist     = '\012\015';
my $letter     = 'a-zA-Z';
my $digit      = '\d';
my $atom_char  = qq{ [$space<>a;\.:\:\\$esc\$ctrl$nonASCII]};
my $atom       = qq{ $atom_char+ };  
my $byte       = qq{ (?: 1?$digit?$digit |
                           2[0-4]$digit |
                           25[0-5] ) };  
my $qtext      = qq{ [$esc$nonASCII$CRlist] };
my $quoted_pair = qq{ $esc ["$nonASCII"] };
my $quoted_str  = qq{ (?: $qtext | $quoted_pair )* " };
my $word       = qq{ (?: $atom | $quoted_str ) };
my $ip_address = qq{ \$byte (?: $dot $byte ) {3} \} };
my $sub_domain = qq{ $letter$digit|0,61] $letter$digit) ;
my $top_level = qq{ (?: $atom_char ){2,4} };
my $domain_name = qq{ (?: $sub_domain $dot) + $top_level };
my $domain = qq{ (?: $domain_name | $ip_address ) };
my $local_part = qq{ $word (?: $dot $word )* };
my $address = qq{ $local_part \$ domain };

return $addr_to_check =~ /$address$/ ox ? $addr_to_check : "");

## Encrypts the password for crosschecking with the username database
sub EncryptPass{

    this encrypts the password

    $PassWord =~ tr/A-Z/a-z/;
    $md5->reset();
    $md5->add($PassWord);
    $PassWord = $md5->hexdigest();
#Purpose: Checks the Username Database for Username and Password
#Pre: Connect information must be defined elsewhere
#Post: $ib is loaded with ib from table if found
sub CheckDB{
    $dbh = DBI->connect ($dsn, $user_name, $password, \%attr);
    # Checks for ib in username table
    $sth = $dbh->prepare(qq{
        SELECT ib FROM username
        WHERE user_name = ? and password = ?
    });
    $sth->execute($User_Name, $PassWord);
    $db_return1 = $sth->fetchrow_hashref();
    $ib = $db_return1->"ib";
    $dbh->disconnect();
}

#Purpose: Gets the User's information for many different databases
#Pre: Databases must be set up correctly
#Post: Data is stored in correct array
sub GetData{
    $dbh = DBI->connect ($dsn, $user_name, $password, \%attr);
    if(!$ib){
        #If logged in
        #Gets all the information about person from brothers table, if in username table
        $sth = $dbh->prepare(qq{
            SELECT * FROM brothers
            WHERE ib = ?
        });
        $sth->execute($ib);
        $db_return1 = $sth->fetchrow_hashref();
        my $field;
        foreach $field (@fields){

```
# Load array with information

# Gets all the information about person from work table, if in username table
$sth = $dbh->prepare(qq{
    SELECT * FROM work
    WHERE ib = ?
});
$sth->execute($ib);
$db_return2 = $sth->fetchrow_hashref();

foreach @field (@WorkFields){
    $work{$field} = $db_return2->{$field};  # Load array with information
}

# Gets all the information about person from Bldata table, if in username table
$sth = $dbh->prepare(qq{
    SELECT * FROM Bldata
    WHERE ib = ?
});
$sth->execute($ib);
$db_return3 = $sth->fetchrow_hashref();
$big = $db_return3->{'big'};
$little = $db_return3->{'little'};

$dbh->disconnect();

# Purpose: Puts all that information back into those many databases!
# Pre: Data arrays must be loaded with information and parsed for errors
# Post: Data is loaded back into databases
sub PutData{
    $dbh = DBI->connect ($dsn, $user_name, $password, \@attr);
    # Must use very long update statement because mysql is hard to work with!!
    if($ib){
        $sth = $dbh->prepare(qq{
            UPDATE brothers SET first_name = ?, last_name = ?,
        });
        $sth->execute($big, $little, $work, $other_info);
    }
}
middle_name = ?, suffix = ?, phone = ?, email = ?,
street1 = ?, street2 = ?, city = ?, state = ?,
init_date = ?, grad = ?
WHERE ib = ?
);  #LOADS THE BROTHERS DATABASE WITH NEW VALUES

$sth = $dbh->prepare(qq{
    SELECT ib FROM work
    WHERE ib = ?
});
$sth->execute($ib);

$db_return1 = $sth->fetchrow_hashref();

#if ib found in work database
if($db_return1->"ib"){
    $sth = $dbh->prepare(qq{
        UPDATE work SET title = ?, work_phone = ?,
    fieldwork = ?, work_address = ?, work_city = ?,
    work_state = ?, work_zip = ?, company = ?
    WHERE ib = ?
});
$sth->execute($work{"title"}, $work{"work_phone"}, $work{"fieldwork"},
$work{"work_address"}, $work{"work_city"}, $work{"work_state"},
$work{"work_zip"}, $work{"company"}, $ib);
      #LOADS WORK TABLE WITH NEW VALUES
} else{
    #if ib not in work database must create entry instead of updating
    $sth = $dbh->prepare(qq{
        INSERT INTO work VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    });
    $sth->execute($ib, $work{"title"}, $work{"work_phone"}, $work{"fieldwork"},
$work{"work_address"}, $work{"work_city"}, $work{"work_state"},
$work{"work_zip"}, $work{"company"});
      #LOADS WORK TABLE WITH NEW VALUES IF NOT ALREADY CREATED LIKE ABOVE

21
## insert big and little into $Ldata table, but first find $L's of last names...

```perl
$sth = $dbh->prepare("SELECT $l FROM $Ldata WHERE $l = ?");
$sth->execute($Id);
$dB_return1 = $sth->fetchRowHashref();
if($dB_return1{"$l"){
  # If $Ldata table has users information already
  $sth = $dbh->prepare("UPDATE $Ldata SET $l = ?, little = ?
WHERE $l = ?");
  $sth->execute($Big, $Little, $Id);
}
else{
  # If $l not in work database must create entry instead of updating
  $sth = $dbh->prepare("INSERT INTO $Ldata VALUES(?,$?,?)");
  $sth->execute($Id, $Big, $Little);
}
$dbh->disconnect();
```

#Purpose: Starts the HTML and prints the title for the page. Also has the JavaScript Embedded in it.
#Pre: None
#Post: Javascript is loaded along with title
sub StartPage{
  print <<<WIODHTML;
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<script src="http://lyco.lycoming.edu/~lcs/js-lib/formLib.js"></script>

function validateForm(form) {
  validateText = new Array("first_name", "last_name", "email");
  var conf = confirm("Are you sure you want to update this information?");
  if(conf == true)
    return requiredValues(form, requiredText) && checkProblems();
```
function checkPassword(form){
  if(form['newpass1'].value != form['newpass2'].value){
    alert("Please retypen your Passwords, they do not match.");
    return false;
  }
  return true;
}

//---
</script>

<p align="center"><font size="7">Iota Beta Zeta<br>
<i>Alumni Database</i></font></p><br>
<p align="center"><u><font size="6">Your Information</font></u></p>

ENDHTML

#purpose: This Prints the Form for User Information
#Pre: Array's of data must be loaded from lca_data database
#Post: form created where user and update his information
sub PrintForm{
  print "<END;"
  <div align="center">
    <center>
      <table border="1" cellspacing="0" cellpadding="0" style="border-collapse: collapse" bgcolor="#111111" width="70%" bgcolor="#9758E4">
        <tr>
          <td width="100%" bgcolor="#000000">
            <form method="POST" action="myinfo.cgi" onsubmit="return validateForm(this);">
              <div align="center">
                <center>
                  <table border="0" style="border-collapse: collapse" bgcolor="#111111" width="90%" cellpadding="2" bgcolor="#9758E4">
                    <tr>
                      <td width="100%" colspan="3" align="center"><b>
<table>
<thead>
<tr>
<th><strong>First Name</strong></th>
<th>&quot;input type=&quot;text&quot; name=&quot;first_name&quot; size=&quot;20&quot; tabindex=&quot;1&quot; value='&quot;$data['first_name']'&quot;&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Last Name</strong></td>
<td>&quot;input type=&quot;text&quot; name=&quot;last_name&quot; size=&quot;20&quot; tabindex=&quot;2&quot; value='&quot;$data['last_name']'&quot;&quot;</td>
</tr>
<tr>
<td><strong>Middle Name</strong></td>
<td>&quot;$data['middle_name']&quot;</td>
</tr>
<tr>
<td><strong>Suffix</strong></td>
<td>&quot;input type=&quot;text&quot; name=&quot;suffix&quot; size=&quot;8&quot; tabindex=&quot;4&quot; value='&quot;$data['suffix']'&quot;&quot;</td>
</tr>
<tr>
<td><strong>Phone</strong></td>
<td>&quot;input type=&quot;text&quot; name=&quot;phone&quot; size=&quot;20&quot; onChange=&quot;checkPhone(this)&quot; tabindex=&quot;5&quot; value='&quot;$data['phone']'&quot;&quot;</td>
</tr>
<tr>
<td><strong>Email</strong></td>
<td>&quot;input type=&quot;text&quot; name=&quot;email&quot; size=&quot;20&quot; onChange=&quot;checkEmail(this)&quot; tabindex=&quot;6&quot; value='&quot;$data['email']'&quot;&quot;</td>
</tr>
<tr>
<td><strong>Current Address</strong></td>
<td>&quot;&lt;p align=&quot;center&quot;&gt;&quot;</td>
</tr>
<tr>
<td><strong>Street</strong></td>
<td>&quot;&lt;input type=&quot;text&quot; name=&quot;street1&quot; size=&quot;29&quot; tabindex=&quot;7&quot; value='&quot;$data['street1']'&quot;&quot;&quot;</td>
</tr>
</tbody>
</table>
| **Line 2** | ""
<table>
<thead>
<tr>
<th>City :</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>State :</td>
<td></td>
</tr>
<tr>
<td>Zip :</td>
<td></td>
</tr>
</tbody>
</table>

If you live outside the US, enter country instead of City and ignore State.

| Initiation Date : |  |

Do not update this information unless you are absolutely certain.

| Graduation Year : |  |

| Big's IB : |  |

| Little's IB : |  |

We would like your work information so other brothers can find jobs through you.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Name</td>
<td></td>
</tr>
<tr>
<td>Job Title</td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td></td>
</tr>
<tr>
<td>Field of Work</td>
<td></td>
</tr>
</tbody>
</table>

- Accounting/Auditing
- Administrative and Support Services
- Advertising/Marketing/Public Relations
- Agriculture, Forestry, & Fishing
- Architectural Services
- Arts, Entertainment, and Media
- Banking
- Biotechnology and Pharmaceutical
- Community, Social Services, and Nonprofit
- Computers, Hardware
- Computers, Software
- Construction, Mining and Trades
<option>Consulting Services</option>
<option>Customer Service and Call Center</option>
<option>Education, Training, and Library</option>
<option>Employment Placement Agencies</option>
<option>Engineering</option>
<option>Finance/Economics</option>
<option>Financial Services</option>
<option>Government and Policy</option>
<option>Healthcare, Other</option>
<option>Hospitality/Tourism</option>
<option>Human Resources</option>
<option>Information Technology</option>
<option>Installation, Maintenance, and Repair</option>
<option>Insurance</option>
<option>Internet/E-Commerce</option>
<option>Law Enforcement, and Security</option>
<option>Legal</option>
<option>Manufacturing and Production</option>
<option>Military</option>
<option>Other</option>
<option>Personal Care and Service</option>
<option>Real Estate</option>
<option>Retail/Wholesale</option>
<option>Sales</option>
<option>Science</option>
<option>Sports and Recreation</option>
<option>Telecommunications</option>
<option>Transportation and Warehousing</option>
<option>Student</option>
</select></td>
</tr>
<tr width="100%" align="right" colspan="3">
<p align="center"><b>Address :<br />
<input type="text" name="work_address" size="33" value="$work["work_address"]"></b></td>
</tr>
<tr width="50%" align="right">
<p align="center"><b>City :<br />
<input type="text" name="work_city" size="20" value="$work["work_city"]"></b></td>
<td width="25%" align="center">State:<br><input type="text" name="work_state" size="3" value='$work{"work_state"}'></td><br><td width="25%" align="center">Zip:<br><input type="text" name="work_zip" onChange="checkZip(this);"
    size="8" value='$work{"work_zip"}'></td>
</tr>
<tr>
    <td width="100%" align="right" bgcolor="#41D526" colspan="3"&nbsp;</td>
</tr>
</table>
</center>
</div>
<p align="center">
<input type="submit" value="Update" name="submit">
<input type="reset" value="Reset Form" name="B2"></p>
</form>
</td>
</td>
</tr>
</table>
</center>
</div>
END

# you can stop reading my code here!
#!/usr/bin/perl

# Should have -T behind, but that was creating an error. If you try this code you should
# try the -T for Taint Checking

# Title: User Name Register Page for LCA Online Database
# Author: Joshua Speicher
# Last Update: 3/20/02
# Purpose: This CGI will allow LCA alumni to register for a username so they can access
# the alumni database of Iota Beta Zeta
# Pre: None
# Post: If User's name found in database and agrees to terms and conditions his info
# username and encrypted password will be inserted into the username table of the
# lca data Database

use DBI; # Database Interface
use CGI; # Common Gateway Interface
use strict; # Strict Subs
use MD5; # Password encryption

my($md5) = new MD5;
my($cgi) = new CGI;

my $title = "Iota-Beta Zeta Alumni Registration";
# All of these variables are checked to have only alphanumeric
my($FirstName) = $cgi->param("first_name") =~ /([\w.]+)\$/; # First name from form
my($LastName) = $cgi->param("last_name") =~ /([\w.]+)\$/; # Last name from form
my($Username) = $cgi->param("username") =~ /([\w.]+)\$/; # Username from form
my($Password1) = $cgi->param("password1") =~ /([\w.]+)\$/; # Password from form
my($Password2) = $cgi->param("password2") =~ /([\w.]+)\$/; # Check Pass Spelling from form
my($CBBox) = $cgi->param("agree_box"); # Check Box from form
my($EmailAddress) = validate_email_address( $cgi->param("email") ); # Validated Email address from form

# START OF DBI SETUP FOR USERNAME AND CONNECTION PARAMS
my($sth);

my($lost_name, $user_name, $password) = (undef, undef, undef);
my($db_name) = "lca_data";
my($dbh,$dsn);
my($attr) = {RaiseError => 1};

$dsn = "dbi:mysql:$db_name";
$dsn .= "@host_name" if $host_name;
$dsn .= ":mysql_read_default_file=/student/stu/1ca/alumni.cnf";
#END OF DBI SETUP

******************************************************************************
print $cgi->header();  #Start HTML for CGI

StartPage();         #Start the page with title and javascript

# Process the form for anything missing, first open, or everything filled in, if
# everything is filled in call CheckForm and FindName functions

if(!$FirstName && !$LastName && !$UserName && !$Pass1 && !$Pass2 && !$CkBox && !$EmailAddress){
    #If nothing filled in on form display the form
    FillForm();
}
elseif(!$FirstName && !$LastName && !$UserName && !$Pass1 && !$Pass2 && !$CkBox && !$EmailAddress){
    #If everything is filled in on the form except username had an illegal character
    print "<center>";
    print $cgi->h2("Username can only have letters, numbers, and underscores.");
    print "</center>";
    FillForm();
}
elseif(!$FirstName || !$LastName || !$UserName || !$Pass1 || !$Pass2 || !$CkBox || !$EmailAddress){
    #If anything missing on the form display error
    print "<center>";
    print $cgi->h2("You must fill in everything requested and agree to the terms and conditions.");
    print "</center>";
    FillForm();
}
else{
    #If everything filled in check the form and update the database
    CheckForm();
    FindName();
}
<head>
<title>Alumni Registration Page</title>
<script src="http://lyco.lycoming.edu/~lca/js-11b/formLib.js"></script>
</head>
<body>
<p align="center"><b><u><font size="6">Welcome to the Iota-Beta Zeta Alumni Registration Page</font></u></b></p>
<p align="center">You will be able to sign up for your username and password to access the Iota Beta Zeta Alumni directory and listings.</p>
</body>
#Purpose: this function checks the form for security breaks and for password mismatches
#Pre: Form Values must be ready in already
#Post: Form fields are validated

sub CheckForm{
    if ($FirstName !~ /\w/){
        print "<center>";
        print $cgi->h2("First Name was not in correct format.");
        print "</center>";
        FillForm();
    }
    if ($LastName !~ /\w/){
        print "<center>";
        print $cgi->h2("Last Name was not in correct format.");
        print "</center>";
        FillForm();
    }
    if ($UserName !~ /\w/){
        print "<center>";
        print $cgi->h2("User Name was not in correct format.");
        print "</center>";
        FillForm();
    }
    if ($Pass1 ne $Pass2){
        print "<center>";
        print $cgi->h2("Rtype Passwords, they do not match.");
        print "</center>";
        FillForm();
    }
}

#Purpose: ThankYou displays a thank you message for registering and redirects to the main page for logging in.
#Pre: None
#Post: Thankyou displayed on screen
#Last Updated: 3/20/02

sub ThankYou{
    print "<END_TEXT;

32
Thank You for registering with Iota-Beta Zeta's Alumni Directory.

An email has been sent to you with your Username and Password.

To log in and continue on with Iota-Beta Zeta's Alumni Directory go to: http://lyco.lycoming.edu/~lca/cgi-bin/alumni/alumni.cgi

---

#Purpose: Email will send a confirmation email to the new user with his username and password
# and instructions on how to login to the LCA database
#Pre: Values must be ready in from form and validated for security issues
#Post: Email sent to user with username and password and instructions on how to login
#Last Updated: 3/20/02

sub Email{
    open MAIL, "| /usr/lib/sendmail -t -i";
    print MAIL <<END_MAIL;
    To: $EmailAddress
    From: IotaBetaZeta@Yahoo.com
    Reply-To: IotaBetaZeta@yahoo.com
    Subject: Iota-Beta Zeta Alumni Registration

    $FirstName $LastName,
    Thankyou for registering with Iota-Beta Zeta's Alumni Database.
    You will be able to find all your old brothers and all the new brothers with this service.

    Please Save this information for later use.
    Your Username is: $UserName
    Your Password is: $Pass1

    To login and search our web page go to http://lyco.lycoming.edu/~lca/cgi-bin/alumni/alumni.cgi

    Thank You,
    From all the Brothers of Iota-Beta Zeta

    P.S. If you have any questions or recieved this email by error send comments to IotaBetaZeta@Yahoo.com
END_MAIL

close MAIL
}

#Purpose: To validate email address for format and security issues
#Pre: Email address must be ready in and passed to function
#Post: Very Safe and correct format email address
#Last Updated: 3/20/02

sub validate_email_address{
    my $addr_to_check = shift;
    $addr_to_check =~ s/(?:["\""]|\[\]|\\|\.)*/\t*/$1/g;

    my $esc    = '\\';
    my $space  = '\040';
    my $ctrl   = '\000-\037';
    my $dot    = '\.';
    my $nonASCII = '\x80-\xff';
    my $CRLlist = '\012\r\n';
    my $letter = 'a-zA-Z';
    my $digit  = '\d';

    my @atom_char = qq{ [^$space<>@,;:\[\]\$esc$ctrl$nonASCII] };
    my $atom    = qq{ $atom_char+ };
    my $byte    = qq{ ([?: 1?$digit?$digit | 2[0-4]$digit | 25[0-5]) });

    my $qtext   = qq{ ["$esc$nonASCII$CRLlist"] };
    my $quoted_pair = qq{ \$esc ["$nonASCII"] );
    my $quoted_str  = qq{ " (?: $qtext | $quoted_pair ) " };;

    my $word    = qq{ (?: $atom | $quoted_str ) };
    my $ip_address = qq{ \$byte \$byte \$byte \$byte \$byte \$byte \$byte \$byte \$byte };
    my $sub_domain = qq{ [$letter$digit! [$letter$digit-][0.61] [$letter$digit$] };
    my $top_level = qq{ (?: $atom_char ) [2,4] };
    my $domain_name = qq{ (?: $sub_domain $dot ) $top_level };
    my $domain = qq{ (?: $domain_name | $ip_address ) };
    my $local_part = qq{ $word (?: $dot $word )* };
    my $address = qq{ $local_part @ $domain };
return $addr_to_check =- /"$address$/ox ? $addr_to_check : ";
)

#Purpose: This will check the database for the name typed in, if not found will post error message
# This will also input the IB, Username, and Encrypted Password into the "username" table
#Pre: Values must be read in and validated
#Post: ib, username, password are inserted into username table
#Last Updated: 3/20/02
sub FindName{
  my($db_return1);
  my($db_return2);
  my($db_return3);
  $dbh = DBI->connect ($dsn, $user_name, $password, \&attr);
  #checks for username in the brothers table
  $sth = $dbh->prepare (qq{
    SELECT ib,first_name, last_name FROM brothers
    WHERE last_name = ? and first_name = ?
  });
  $sth->execute ($LastName, $FirstName);
  #Get everything from brothers table
  $db_return1 = $sth->fetchrow_hashref();
  #Fetch row from database return
  my($ib) = $db_return1->{ib};
  #Set Ib to ib from table
  # Checks for ib in username table, can't have two of the same usernames
  $sth = $dbh->prepare (qq{
    SELECT ib FROM username
    WHERE user_name = ?
  });
  $sth->execute ($UserName);
  #Check if username already registered
  $db_return2 = $sth->fetchrow_hashref();
  #Fetch the database return
# Checks for $ib in username table, if there user has already registered

$sth = $dbh->prepare(qq{
    SELECT ib FROM username
    WHERE ib = ?
});
$sth->execute([$ib]); # Execute with $ib
$db_return3 = $sth->fetchrow_hashref(); # Fetch results

if($db_return3->{ib}){
    # If $ib already registered display error message
    print "<center>
    print $cgi->h2("You have Already Registered!");
    print "</center>
    print "<hr>
    print "<p align="center">To log in and continue on with Iota-Beta Zeta's Alumni Directory go <a href="http://lyco.lycoming.edu/~lca/cgi-bin/alumni/alumni.cgi">here</a>.</p>";
}
elseif($db_return2->{ib}){
    # If username already registered display error message
    print "<center>
    print $cgi->h2("Username in Database already. please try another");
    print "</center>
}
elseif($db_return1->{first_name} && !$db_return2->{ib}) # If $ib in brother's table and username not used
{
    $sth = $dbh->prepare(qq{
        INSERT INTO username
        values (?, ?, ?)
    });

    ###############################################################################
    # This encrypts the password ###############################################################################

    $Pass1 = $Pass1/1/a-z/; # This Enters the $ib, Username, and encrypted password into the username table
    $Pass1 = md5($Pass1);
    my $encrypted = $md5->hexdigest();
    ###############################################################################

    $sth->execute([$ib, $UserName, $encrypted]);
# This code will enter the updated email address into the brothers table
riculum:

    $sth = $dbh->prepare("UPDATE brothers SET email = ? WHERE ib = ?

);

    $sth->execute($EmailAddress, $iB);

    # Email User his information and password
    Email();
    # Print Thankyou and redirect to another page
    ThankYou();
}

elsif (!$db_return1->[first_name]) {
    # If not a brother of iota-beta zeta or misspelled name
    print "<center>";
    print $cgi->h2("Your name was not found in our database, make sure you use your full name.");
    print "</center>";
    # Finish page if error
    FillForm();
}

$dbh->disconnect();

# Close DBI interface

# Purpose: This will post the Form to be filled in by the User, it will repost data previously typed in if there was an Error
# Pre: None
# Post: None
# Last Updated: 3/20/02
sub FillForm

print "<HR";


<p align="center">The data stored on the Iota-Beta Zeta Alumni page is to be used and viewed only by alumni of Iota Beta Zeta.</p>

<p align="center">Nobody will be allowed to modify or change your information without your strict agreement.</p>

<p align="center">Your information will be kept private for the use of Iota-Beta
Zeta and its Alumni only.

Check this box if you agree to the terms above.

First Name

Last Name

Username

Email Address

Password
<tr>
<td width="50%" align="right"><font size="4">Retype Password</font></td>
<td width="50%" align="left">
<input type="password" name="password2" size="20" tabindex="6"></td>
</tr>
</table>
<p align="center"><input type="submit" value="Submit" name="Submit"><input type="reset" value="Reset" name="Reset"></p>
</form>
<p align="center">&nbsp;</p>

ENDTEXT
}

print $cgi->end_html(); #Finish off HTML
use DBI;
use CGI;
use strict;
use MD5;

my($md5) = new MD5; # Used for encrypting
my($cgi) = new CGI; # Used for using CGI script

my($UserName) = $cgi->cookie("user_name") =~ /\b([\w.]+)\b/; # Gets username from Cookie
my($PassWord) = $cgi->cookie("password") =~ /\b([\w.]+)\b/; # Gets Password from Cookie
my($Function) = $cgi->param("func") =~ /\b([\w.]+)\b/; # Gets Function Parameter from html space
my($$SearchValue) = $cgi->param("keyword") =~ /\b([\w.]+)\b/; # Match only alphanumeric and spaces
my($$Select) = $cgi->param("select") =~ /\b([\w.]+)\b/; # Match Only alphanumeric
my($$ID); # the users ID stored in one easy to access variable
my($$DB_return); # Used to fetch data from MySQL
my ($$For) = $cgi->param("for") =~ /\b([\w.]+)\b/; # the search delimiter, like a letter or name
my ($$By) = $cgi->param("cat") =~ /\b([\w.]+)\b/; # What to search for
my ($$Order) = $cgi->param("ord") =~ /\b([\w.]+)\b/; # What to search for
if(!$Order){
    $Order = "ib";
}

# START OF DBI SETUP FOR USERNAME AND CONNECTION PARAMS
my($sth);

my($host_name, $user_name, $passsword) = (undef, undef, undef);
my($db_name) = "lca_data";
my($dbh,$dsn);
my($attr) = (RaiseError => 1);

$dsn = "dbi:mysql:$db_name";
$dsn .= ":host_name=$host_name" if $host_name;
$dsn .= ":mysql_read_default_file=/student/stu/lca/alumni.cnf";
#END OF DBI SETUP

#Print Header
print $cgi->header(-type => "text/html");
print $cgi->start_html(-title => "Iota-Beta Zeta Search");

CheckDB();
if($ID)
  StartPage();
  if($SearchValue){
    Find();
  } elsif(($Function eq "search")
    Search();
  } elsif(($Function eq "browse")
    Browse();
  } elsif(($Function eq "lookat")
    LookAt();
  } else{
    PrintForm();
  }
else{
  #if error do this. print error message
  print $cgi->center,
    $cgi->h1("ERROR!"),
    $cgi->h1("You are not logged in or cookies are not enabled, please return to the main alumni page.");
}

#this prints the links to the other Alumni Pages
print "</table>\n";

41
# Purpose: Checks email address for correct format
# Pre: none
# Post: Checked email address without any "dangerous" content
sub validate_email_address{
    my $addr_to_check = shift;
    $addr_to_check =~ s/(^[^@]*@[^@]*@[^@]*$)/1/g;
    my $esc   = '\\\';
    my $space = '040';
    my $ctrl  = '037';
    my $dot   = '\.';
    my $nonASCII = '\x80-\xff';
    my $CRlist = '\012\015';
    my $letter = 'a-zA-Z';
    my $digit = '\d';
    my $atom_char = qq{[^$space@$,:;\.|\.]|\$esc$ctrl$nonASCII}|;
    my $atom = qq{ $atom_char+ };    
    my $text = qq{[$esc$nonASCII$CRlist]*};    
    my $quoted_pair = qq{ $esc ["\$nonASCII] };    
    my $quoted_str = qq{ " (?: $text | $quoted_pair )* " };    
    my $word = qq{ (?: $atom | $quoted_str ) };    
    my $ip_address = qq{ \$byte (?: $dot $byte ) [3] \} };
    my $sub_domain = qq{ [$letter$digit] };
}
my $stop_level = qq{ (?: $atom_char ){2,4} };  
my $domain_name = qq{ (?: $sub_domain $dot )* $stop_level }; 
my $domain = qq{ (?: $domain_name | $ip_address ) };  
my $local_part = qq{ $word (?: $dot $word )* };  
my $address = qq{ $local_part \@ $domain* }; 

return $addr_to_check =~ /"$address$/ox ? $addr_to_check : "";

#Purpose: To encrypt the password with 128 Bit encryption  
#Pre: MD5 Module must be installed and working and $Password must be set  
#Post: Sets Password to encrypted hex string  
sub EncryptPass{
  ############### this encrypts the password ###############  
$PassWord =~ tr/A-Z/a-z/;  #Sets to all lowercase  
$md5->reset();  
$md5->add($PassWord);  
$PassWord = $md5->hexdigest();  
}

#Purpose: Checks the Username Database for Username and Password  
#Pre: username table must be defined with username and encrypted password  
#Post: returns ib of user if user is logged in  
sub CheckKDS{
  $dbh = DBI->connect($dsn,User_name, $password, \%attr);  
  # Checks for ib in username table  
  $sth = $dbh->prepare(qq{
    SELECT ib FROM username  
    WHERE user_name = ? and password = ?  
  });  
  $sth->execute($UserName, $PassWord);  
  $db_return1 = $sth->fetchrow_hashref();  
  $ib = $db_return1->("ib");  #Sets $ib to ib found in username table  
  $dbh->disconnect();  #Disconnects from mysql DBI
#Purpose: To search either contents on page or entire database for desired information
#Pre: Alumni Database must be created
#Post: The search results will be displayed on a page in table form with ib, firstname, lastname
sub Find{
    my($data);
    # hash of search return
    my(@Results);
    # Array of hashed search results
    if(!_$SearchValue){    # If no searchvalue set to searchvalue from browse commands
        $SearchValue = $For;
    }  
    if(!_$Select){  # If no select set to by value from browse commands
        $Select = $By;
    }
    $dbh = DBI->connect ($dsn, $user_name, $password, \%attr); 
    $Function = "search";
    # Make sure function is set to Keyword for links
    $For = $SearchValue;  # Set $For to searchvalue for links
    $By = $Select;  # Set $By to select for hyperlinks in results
    # search parameter
    if($Select eq "ib"){   # If searching for specific ib
        $sth = $dbh->prepare(qq{
            SELECT ib, first_name, last_name
            FROM brothers
            WHERE ib = ?
        });
        $sth->execute($SearchValue);
    } else{  # If searching for anything else in database
        $sth = $dbh->prepare(qq{
            SELECT ib, first_name, last_name
            FROM brothers
            WHERE $Select LIKE ?
            ORDER BY $Order
        });
        my $temp = "$SearchValue$";
        $sth->execute($temp);
    }
    while($data = $sth->fetchrow_hashref()){  # While still more results
push(@Results, $data);  #Push onto results array

$sth->finish();  #Make sure DBI is finished

if(@Results){
  Results(@Results);
  #Print the Table of Results
}
else{  #If no results
  print $cgi->center,
  $cgi->h3("Search returned no results");
  print "<a href="search.cgi">"<font color="#000000" size="4">Main Page</a>
<br><hr>
;
}

$dbh->disconnect();  #Disconnect from DBI

#Purpose: Search the databases for desired information and display in nice table format with links
#
To get desired information on individual Alumni
#Pre: Alumni database must be created with desired information
#Post: Information displayed on the screen in neat and orderly fashion
sub Search{

  $dbh = DBI->connect ($dsn, $user_name, $password, \%attr);

  my ($data1);  #holds data from table for display

  if($ib){  #Make sure user is logged in, and not just trying to get around security features
    my ($For) = $cgi->param("for") = - /^(\w+\s*\.|\s\.|\s)/;  #the search delimiter, like a letter or name
    my ($By) = $cgi->param("cat") = - /^(\w+)$/;  #What to search for
    my ($Order) = $cgi->param("ord") = - /^(\w+)$/;  #What to search for
    if(!$Order){
      $Order = "ib";
    }
    my (@Results);  #Array of search results

    #both of these searches would access brothers table for first and last name
    if($By eq "fieldofwork") || ($By eq "work_city") || ($By eq "work_state"){
      #If searching in work table for work information
$sth = $dbh->prepare(qq{
    SELECT brothers.ib, first_name, last_name
    FROM work, brothers
    WHERE brothers.ib = work.ib
    AND $By = ?
    ORDER BY $Order
});
$sth->execute($For);
while($data1 = $sth->fetchrow_hashref()){
    #While more results from search
    push(@results, $data1);
    #Push onto results array
}
$sth->finish();
#Finish up DBI Work
$dbh->disconnect();
#Disconnect from DBI
}
elsif(($By eq "last_name") || ($By eq "first_name")){
    #if Searching for Last Name or First Name
    $sth = $dbh->prepare(qq{
        SELECT ib, first_name, last_name
        FROM brothers
        WHERE $By Like ?
        ORDER BY $Order
    });
    my $temp = "$For%";
    #Put "%" at end of search string to
    # indicate anything else...so if your searching
    # for all names starting with S
    $sth->execute($temp);
    while($data1 = $sth->fetchrow_hashref()){
        #If more results
        push(@results, $data1);
        #Push onto results array
    }
    $sth->finish();
    #Finish DBI work
    $dbh->disconnect();
}
else{
    #if searching in brothers table for brother information
    $sth = $dbh->prepare(qq{
        SELECT ib, first_name, last_name
        FROM brothers
        WHERE $By = ?
        ORDER BY $Order
    });
    my $temp = "$For%";
    #Put "%" at end of search string to
    # indicate anything else...so if your searching
    # for all names starting with S
    $sth->execute($temp);
    while($data1 = $sth->fetchrow_hashref()){
        #If more results
        push(@results, $data1);
        #Push onto results array
    }
    $sth->finish();
    #Finish DBI work
    $dbh->disconnect();
}


```php

nett

if($Results){
    Results(@Results);
}
else{
    print $cgi->center,
    $cgi->h3("Search returned no results");
    print "<a href="/search.cgi"><font color="#00000" size="4">Main Page</a></b></a><br><hr><br>
}

#Purpose: To print the table of results from searches
#Pre: Must have hash of results from table queries
#Post: I displayed table of ib, first name, last name, with hyperlink
sub Results{
    my $element;

    print "<br>";
    print "\n\center\n";
    print "<font size="5">Results</font>";
    #Print start of table and table heading Titles
    print "<table border="1" cellspacing="0" cellpadding="21" width="75%" bgcolor="#111111" bordercolor="#111111">";
    <tr align="center"><font size="5">
        <a href="/search.cgi?func=$Function&for=$For&ord=ib&cat=$By">IB</a></td>
        <a href="/search.cgi?func=$Function&for=$For&ord=first_name&cat=$By">First Name</a></td>
      </tr>";
    print "\n\</table>";
}
```
while($element = shift){
    print "<tr><td align="center">
    <a href="search.cgi?func=For&ord=last_name&cat=By">
        Last Name</a></td>
    </tr>";
    print "<tr><td align="center">
    <a href="search.cgi?func=lookat&for=$element->{ib}">
        $element->{ib}</a></td>
    </tr>";
    print "<tr><td align="center">
    <a href="search.cgi?func=lookat&for=$element->{first_name}">
        $element->{first_name}</a></td>
    </tr>";
    print "<tr><td align="center">
    <a href="search.cgi?func=lookat&for=$element->{last_name}">
        $element->{last_name}</a></td>
    </tr>";
    print "</table>";
    print "\n</center>";
    print "\nnbsp";  
}

#Purpose: to be able to Browse by different catagories, needs to access database,
# very sorry for the messy nature, will try to clean up later
# This Function is very long because it has many "if" statements, please read carefully
#Pre: Database must be created with tables and information
#Post: links to different search parameters organized neatly for easy browsing
sub Browse{
    my($Cat) = $cgi->param("for");
    #Get what to search for

    $dbh = DBI->connect ($dsn, $user_name, $password, \$attr);

    if($Idb){
        #If logged in to the alumni database site
        #Browse by Different Catagories, each one requiring it's own function
        if($Cat eq "first_name"){
            #Browse by first name
            print "\n<br>";
            print "\n</center>");
            print "<font size="5">Browse By</font>";  #Print Title
            print $cgi->h3("First Name");
            AlphabetTable("first_name");
            print "\n</center>";
            print "\nnbsp";
        }
    }
    elsif($Cat eq "last_name"){
        #Browse by Last Name
        print "\n</center>");
        print "<font size="5">Browse By</font>";
    }
print cgi->h3("Last Name");
AlphabetTable("last_name");  #Show alphabet table with last_name as links
print "\n</center>");
print "&nbsp;";
}

else if($Cat eq "Grad"){
  #Browse by Graduation Date
  $sth = $dbh->prepare(qq{
    SELECT distinct grad
    FROM brothers
    ORDER BY grad
  });
  $sth->execute();  #Get all the distinct graduation dates from the database

  print "\n<center>";
  print "<font size="5">Browse By<font>";
  print cgi->h3("Graduation Year");
  #While there are more graduation dates
  while($db_return1 = $sth->fetchrow_hashref()){
    #Print the table of graduation dates
    print "<font size="4">
      <a href="search.cgi?func=search&for=$db_return1->{grad}&cat=grad">$db_return1->{grad}</a></font>";
  }
  print "\n</center>");
  print "&nbsp;";
}

else if($Cat eq "init_date"){
  #Browse by Initiation Date
  $sth = $dbh->prepare(qq{
    SELECT distinct init_date
    FROM brothers
    ORDER BY init_date
  });
  $sth->execute();  #Get all the distinct initiation dates from the database

  print "\n<center>";
  print "<font size="5">Browse By<font>";
  print cgi->h3("Initiation Date");  #Print title of results
  print "<table border="1" cellpadding="2" cellspacing="0" width="\$5%"
    bordercolor="#111111" width="\$5%"
    bgcolor="#101010"">";
  my $count = -1;  #Set up counter
while($db_return1 = $sth->fetchrow_hashref()){
    $count = $count + 1;  # Increment counter for # of cells in row
    if($count > 4){
        $count = 0;  # Reset counter because starting new row
        print "<tr><td align="center"><font size="4">
";  # New row
        print <a href="search.cgi?func=search&for=$db_return1-
>init_date)&cat=init_date"/>
";  # URL
        print "$db_return1->[init_date]";
    }
}
else{
    $count = 0;  # Under 5 elements in row
    print "<tr><td align="center"><font size="4">
";  # New row
    print <a href="search.cgi?func=search&for=$db_return1-
>init_date)&cat=init_date"/>
";  # URL
    print "$db_return1->[init_date]";
}
}
print "</table>";
print "</center>";
print "&nbsp;";
}
elsif($Cat eq "city"){
    # Browse by City
    $sth = $dbh->prepare(qq{
        SELECT distinct city, state
        FROM brothers
        ORDER BY state
    });
    $sth->execute();  # Get all the distinct city, state combinations from the database
    print "<center>";
    print "<font size="5">Browse By</font>";
    print "$cgi->h3("City")";  # Print title of results
    print "<table border="1" cellpadding="2" cellspacing="0" width="90%" bgcolor="#41D526">";  # Table
    my $count = -2;  # Set counter for # of cells in a row
    while($db_return1 = $sth->fetchrow_hashref()){
        $count = $count + 1;  # Increment Counter
        if($db_return1->{city}){  # If element to display
            if($count > 3){  # If 4 elements in row
                $count = 0;  # Reset counter and start a new row
                print "<tr><td align="center"><font size="4">
";  # New row
            }
        }
    }
```html
<a href="search.cgi?func=search&for=$db_return1-
{city}&cat=city">"</a>

{state}</a></font></td>";
}
</td>
} else{
    #if under 4 elements in row
    print "<td align="center"><font size="4">
    <a href="search.cgi?func=search&for=$db_return1-
{city}&cat=city">"</a>
    </font></td>
</font></td>";
}
</table>;
print "\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n
```
} else { # Under 5 elements in row print "<td align="center">"<font size="4">
  <a href="search.cgi?func=search&for=$db_return1-}
  <state>$state</state>&cat=state"></a></font></td>;
}

print "</table>
print "
</center>
print "&nbsp;"

} elsif($Cat eq "fieldofwork"){
  # Browse by Field of Work
  $sth = $dbh->prepare(qq{
    SELECT distinct fieldofwork
    FROM work
    ORDER BY fieldofwork
  });
  $sth->execute(); # Select all the distinct fields of work from the database
  print "
</center>
print "<font size="5">"-Browse By</font>"
print "table border="1" cellpadding="2" cellspacing="0""
bordercolor="#111111" width="75%" bgcolor="#41D526">
my $count = -1; # Initialize cell counter
while($db_return1 = $sth->fetchrow_hashref()){
  # If more data from database
  $count = $count + 1; # Increment cell counter
  if($db_return1-}{fieldofwork}{
    # Only 3 elements per row
    if($count > 2){
      $count = 0; # Reset cell counter because starting new row
      print "<tr><td align="center">"<font size="4">
        <a href="search.cgi?func=search&for=$db_return1-}{fieldofwork}&cat=fieldofwork"></a></font></td>;
    }
  }
  else{
    # Still under 3 elements in the row
    print "<td align="center">"<font size="4">
      <a href="search.cgi?func=search&for=$db_return1-
      {fieldofwork}&cat=fieldofwork"></a></font></td>;
  }
}
print "</table>
print "\n</center>
print "&nbsp;"

} elseif($Cat eq "work_city"){
    #Browse by Work City
    $sth = $dbh->prepare(qq{
        SELECT distinct work_city, work_state
        FROM work
        ORDER BY work_state
    });
    $sth->execute(); #Select all the distinct work_city, state combinations from the database

    print "\n</center>
    print "<font size="5">"Browse By</font>"
    print $qi->h3( "Work City" ); #Print title of search results
    print "<table border="1" cellpadding="2" cellspacing="0" background="#111111" width="75%" bgcolor="#41D526">";
    my $count = 2; #initialize cell counter
    while($db_return1 = $sth->fetchrow_hashref()){
        if more cells to display
        $count = $count + 1; #Increment cell counter
    if($db_return1->{work_city}){
        #If not null
        if($count > 3){ #Only 4 elements per row of results
            $count = 0; #Reset cell counter because start new row
            print "<tr>
            <td align="center"><font size="4">"<a href="search.cgi?func=search&for="$db_return1- 
            >>{work_city}'&cat=work_city\">
            >{work_state}</a></font></td>
        }
    else{
        #Under 4 elements in the row
        print "<tr align="center"><font size="4">
        <a href="search.cgi?func=search&for="$db_return1- 
        >>{work_city}'&cat=work_city\">
        >{work_state}</a></font></td>
    }

    }
print "</table>";
print "\n\n</center>";
print "&nbsp;"
}

else{
  # Browsing by Work State
  $sth = $dbh->prepare(qq{
    SELECT distinct work_state
    FROM work
    ORDER BY work_state
  });
  $sch->execute();
  print "\n\n</center>";
  print $sch->get_result()->Num();
  print $cgi->h3("Work State");
  print $cgi->h3("Browse By");
  print $cgi->h3("Search Results");
  print "<table border="1" cellspacing="0" width="75%" background="#41B5E6">";
  my $count = -2;
  if($dbmNeed1 = $sth->fetchrow_hashref()){
    if(count > 3)
      " Only 4 elements per row"
    else{
      print "<tr align="center"><td align="center">";
      print $dbmNeed1-></a>
      "$work_state";";
    }
  }
  print "</table>";
print "\n\n</center>";
print "&nbsp;";
}
# Purpose: To print the alphabet table with either first_name or last_name as search criteria
# Pre: What browsing for must be past to function...either first_name or last_name
# Post: a table of all alphabet with hyperlinks to search for that letter and type

sub AlphabetTable{
    my $name = shift;
    # Get type from passed value
    print "table border="1" cellpadding="2" cellspacing="0"
bordercolor="#111111" width="75%"bgcolor="#41D526">
    print <<END;
<tr>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=ACat=$name">A</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=BCat=$name">B</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=CCat=$name">C</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=DCat=$name">D</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=ECat=$name">E</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=FCat=$name">F</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=GCat=$name">G</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=HCat=$name">H</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=ICat=$name">I</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=JCat=$name">J</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=KCat=$name">K</a></font></td>
<td width="7%" align="center"<font size="4">
<a href="search.cgi?func=searchfor=LCat=$name">L</a></font></td>
</tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=M&cat=$name">M</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=N&cat=$name">N</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=O&cat=$name">O</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=P&cat=$name">P</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=Q&cat=$name">Q</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=R&cat=$name">R</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=S&cat=$name">S</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=T&cat=$name">T</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=U&cat=$name">U</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=V&cat=$name">V</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=W&cat=$name">W</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=X&cat=$name">X</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=Y&cat=$name">Y</a></td></tr>
<tr>
<td width="8%" align="center"><a href="search.cgi?func=search&for=Z&cat=$name">Z</a></td></tr>
</table>

END

}
print <<ENDHTML;  #Start HTML printing
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script src="http://lyco.lycoming.edu/1cs/js-lib/formLib.js"></script>
</script>
<p align="center"><font size="7">Iota-Beta Zeta<br>
<i>Search</i></font></p>
<Div align="center">
<Center>
<table border="1" cellspacing="0" cellpadding="0" bgcolor="#F0F8FF" style="border-collapse: collapse" bordercolor="#111111" width="70%" id="AutoNumber1" $
<tr>
<td width="100%" bgcolor="#000000">
<form method="POST" action="search.cgi">
<p align="center"><b></b></p>
<Keyword Search/><br/>
<input type="hidden" name="func" value=$Function>
<input type="hidden" name="for" value=$For>
<input type="hidden" name="cat" value=$By>
<input type="text" name="keyword" size="20"/>
<br/>
<input type="submit" value="Search" name="submit"><br/>
<input type="radio" name="select" value="last_name" checked>Last Name
<input type="radio" name="select" value="first_name" name="select">First Name
<input type="radio" name="select" value="city">City
<input type="radio" name="select" value="state">State
<input type="radio" name="select" value="ib">IB
<br/>
</form>
</Center>
</Div>
</p>
ENDHTML
;

#Purpose: Prints the Form for browsing and searching
#Pre: None
#Post: table for browsing is displayed on screen
sub PrintForm{
End all DBI work

# Set data to hash of family tree information
my $data3 = $dbh->fetchrow_hashref();

# Set data to hash of work information
my $data2 = $dbh->fetchrow_hashref();

# Set data to hash of brothers table information
my $data1 = $dbh->fetchrow_hashref();

# Checks for ib in username table
$sdbi = $dbh->connect($dsn, $user_name, $password, \'wattr\');
$sdbi->prepare("SELECT * FROM WORK WHERE ib = ?");
$sth = $sdbi->prepare("SELECT * FROM BLDATA WHERE ib = ?");
$sth = $sdbi->prepare("SELECT * FROM SBRDATA WHERE ib = ?");
$sth = $sdbi->prepare("SELECT * FROM SBRDATA WHERE ib = ?");
<table>
<thead>
<tr>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Name</strong></td>
</tr>
<tr>
<td><strong>Middle Name</strong></td>
</tr>
<tr>
<td><strong>Last Name</strong></td>
</tr>
<tr>
<td><strong>Phone Number</strong></td>
</tr>
<tr>
<td><strong>Email Address</strong></td>
</tr>
<tr>
<td><strong>Current Address</strong></td>
</tr>
<tr>
<td><strong>Street 1</strong></td>
</tr>
<tr>
<td><strong>Street 2</strong></td>
</tr>
<tr>
<td><strong>City</strong></td>
</tr>
<tr>
<td><strong>State</strong></td>
</tr>
</tbody>
</table>
<p align="center">Zip :&nbsp;<font color="#ffff00">$data->{zip}</font></p></td></tr></td></tr>
<tr><td bordercolor="#41d526" align="right" width="100%" bgcolor="#41d526" colspan="3">Lambda Chi Alpha Information</td></tr></table>

<table>
<thead>
<tr>
<th>Initiative Date</th>
<th>Graduation Year</th>
<th>Big's IB</th>
<th>Little's IB</th>
<th>Work Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>$data-&gt;{init_date}</td>
<td>$data-&gt;{grad}</td>
<td>$data-&gt;{big}</td>
<td>$data-&gt;{little}</td>
<td>&lt;p align=&quot;center&quot;&gt;Company Name :&lt;br&gt;&lt;font color=&quot;#ffff00&quot;&gt;$data2-&gt;{company}&lt;/font&gt;&lt;/p&gt;&lt;p align=&quot;center&quot;&gt;Job Title :&lt;br&gt;&lt;font color=&quot;#ffff00&quot;&gt;$data2-&gt;{title}&lt;/font&gt;&lt;/p&gt;&lt;p align=&quot;center&quot;&gt;Phone :&lt;br&gt;&lt;font color=&quot;#ffff00&quot;&gt;$data2-&gt;{work_phone}&lt;/font&gt;&lt;/p&gt;</td>
</tr>
<tr>
<td>Address</td>
<td>City</td>
<td>State</td>
<td>Zip</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>$data2-&gt;{work_address}$</td>
<td>$data2-&gt;{work_city}$</td>
<td>$data2-&gt;{work_state}$</td>
<td>$data2-&gt;{work_zip}$</td>
<td></td>
</tr>
</tbody>
</table>

# End HTML Printing

# you can stop reading my code here!
//formLib.js
//Common functions used with data checking in forms
//The Following functions are adaptations from CGI Programming with Perl
// Function Adaptation by Joshua Speicher
// Last Updated 4/2/02

//Used as a hash to track element problems
validate = new Object();

//Used to check if value is an integer
function isInteger(value){
    return(value == parseInt(value));
}

//Checks a range for correctness
function inRange(value, low, high){
    return(! (value<low) && value<high);
}

//checks values against certain formats such as '#####', '####-####-####'
function checkFormat(value, format){
    if(value.length != format.length){
        return false;
    }
    for(var i = 0; i < format.length; i++){
        if(format.charAt(i) == '#' && !isInteger(value.charAt(i))){
            return false;
        }
        else if(format.charAt(i) != '#' && format.charAt(i) != value.charAt(i)){
            return false;
        }
    }
    return true;
}

//takes a form and an array of element names, then verifies that each has a value
function requiredValues(form, requiredValues){
    for(var i = 0; i < requiredValues.length; i++){
element = requiredText[i];
if(form[element].value == ""){
    alert("Please enter a value for " + element + ".");
    return false;
}
return true;

// Takes a form and an array of element names, verifies that each has an option
// selected. Assumes the first element is instructions so skips first element
function requiredSelects(form, requiredSelect){
    for(var i = 0; i < requiredSelect.length; i++){
        element = requiredSelect[i];
        if(form[element].selectedIndex <= 0){
            alert("Please select a value for " + element + ".");
            return false;
        }
    }
    return true;
}

// Takes a form and an array of element names; verifies that each has a value checked
function requiredRadios(form, requiredRadio){
    for(var i = 0; i < requiredRadio.length; i++){
        element = requiredRadio[i];
        isChecked = 0;
        if(form[element].checked){
            isChecked = 1;
        }
        if(isChecked == 0){
            alert("You Must Agree to the Terms");
            return false;
        }
    }
    return true;
}

// Verify there are no uncorrected formatting problems with elements
// validated on a per element basis
function checkProblems() {
    for (element in validate) {
        if (!validate[element]) {
            alert("Please correct the format of " + element + ".");
            return false;
        }
    }
    return true;
}

// Verifies that the value of the provided element has # format
function checkStr(element) {
    if (checkFormat(element.value, "#")) {
        alert("Please enter a five digit zip code.");
        element.focus();
        validate[element.name] = false;
    } else {
        validate[element.name] = true;
    }
    return validate[element.name];
}

// Verifies that the Email Address is in the format something@something.com
function checkEmail(element) {
    at = element.value.indexOf("@");
    at2 = element.value.indexOf(".");
    if (at < 0 || at < at2) {
        alert("Must have a Valid Email Address.");
        element.focus();
        validate[element.name] = false;
    } else {
        validate[element.name] = true;
    }
    return validate[element.name];
}

// Checks that the date is in this format: ####
function checkDate(element) {
if(!checkFormat(element.value, "####")){
    alert("Please enter the year of Graduation as ####.");
    element.focus();
    validate[element.name] = false;
} else{
    validate[element.name] = true;
} return validate[element.name];

//Makes sure the initiation date is in the format YYYY-MM-DD
function checkInit(element){
    if(!checkFormat(element.value, "###-###-###")){
        alert("Please enter Date of Initiation as YYYY-MM-DD.");
        element.focus();
        validate[element.name] = false;
    } else{
        validate[element.name] = true;
    } return validate[element.name];
}

//Checks to make sure the ib is all numbers
function checkIB(element){
    if(!isInteger(element.value)){
        alert("The Ib must be all Numbers.");
        element.focus();
        validate[element.name] = false;
    } else{
        validate[element.name] = true;
    } return validate[element.name];
}

//Checks to see if phone number is in correct format
function checkPhone(element){
    if(checkFormat(element.value, "(###) ###-####") || checkFormat(element.value, "# # # -# # # #")){
        validate[element.name] = true;
    } else{
        alert("Invalid Phone Number");
        element.focus();
        validate[element.name] = false;
    } return validate[element.name];
}
else{
    alert("Please enter phone number in this format (###)###-####.");
    element.focus();
    validate[element.name] = false;
}
return validate[element.name];

// Makes sure the zip code is in the correct format and all numbers
function checkZip(element){
    if(checkFormat(element.value, "####") | checkFormat(element.value, "####-####")){
        validate[element.name] = true;
    }else{
        alert("Please the Zip Code as 5 digit or extended.");
        element.focus();
        validate[element.name] = false;
    }
return validate[element.name];
}