1 / VOL. 3, NO. 6
THE CHEMICAL EDUCATOR
© 1998 SPRINGER-VERLAG NEW YORK, INC.

ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

Computers in Chemistry

A Series of CGI/Perl Scripts for Web-Based Feedback and Reporting in the General Chemistry Laboratory: Colorimetry

JOSEPH F. LOMAX*, AND DEBRA K. DILLNER
United States Naval Academy
Annapolis, MD 21402-5026
lomax@brass.mathsci.usna.edu
dillner@brass.mathsci.usna.edu

JOHN W. VERDE
Johns Hopkins University
Baltimore MD 21218
jverde@extremeweb.com

The series of CGI scripts for the experiment “Analysis of Brass” provide the student with prompt, useful feedback on calculations and analyses.

A series of CGI/Perl scripts for a colorimetry laboratory report have been written, implemented and tested. These scripts assist the students by providing feedback on their data handling and analysis. Once the students have learned how to correctly perform the analyses, their results are submitted to their instructor, who can use the output for evaluation. Student assessment of this series of scripts is overwhelming positive. Scripts are included and programming considerations are discussed.

Introduction

Computers should be used to help us do what we have

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

difficulty doing well. We should not have them automate processes that we already do well at a low cost. For instance, automation of a 10-student senior-level special topics course on the current state of biotechnology would be costly in both time and money with few outweighing benefits. One place computers can be used efficiently is in large classes to extract, manipulate, and report data. In addition, with the advent of the Internet, computers can communicate over any distance at any time [1]. Though computers have been used to teach students by presenting models in 3D [2–6] and by showing how multiple variables relate to each other [7], this article will explore assisting the student with analysis for the laboratory report and assisting the instructor in grading the final report by utilizing the computer's speed and accessibility.

One essential component to a chemistry class is the laboratory, and for every experiment there is a report. Most reports have some computation and analysis and, for the instructor, there is little that is more frustrating than finding a computational error that leads the student to an incorrect final analysis. The student, for his or her part, is oblivious to this error until the laboratory report is returned a week or more later. The feedback is so removed in time from the original analysis, they can neither learn from their mistakes nor receive credit for learning. It is part of the learning process to discover the source of the error and to correct it; however, a student is not able to do so without the knowledge that their analysis or computation is incorrect. With the CGI scripts described below, the student can get quick feedback on numerical errors and can be expected to correct them.

Finally, part of the instructor's job, whether they like it or not, is evaluation. Any help that an instructor can get for making evaluation of the student's performance more accurate and quicker, is sure to be appreciated. These CGI scripts are also designed to help in the evaluation process.

The Experiment: Analysis of Brass

Two decades ago, while Professor Ed Koubek of our Department was in a machine shop, he noticed the brass fillings in a mound on the floor and thought that he could put together a colorimetric experiment to analyze the copper content. The experiment and write-up for "Analysis of Brass" (exp7a.PDF) from the General Chemistry Laboratory Manual is available in the supporting material (36jl1897.zip). The experiment involves taking a brass sample, manipulating it chemically to form a copper(II) complex, and

ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

using the absorbance of visible light at 550 nm as a colorimetric probe. This wavelength is used because (1) the copper(II) complex absorbs there, (2) there is no significant interference from other species in the solution, and (3) the instrumentation (Spectronic 20 visible spectrometer) is simple and inexpensive. As is common in many colorimetric analyses, quantifying the copper in “Analysis of Brass” requires: chemical manipulation and amplification, calibration of the spectrometer with standards of known concentration, and comparison of absorption of the unknown solution to the calibration.

1. The chemical manipulation and amplification of the copper are performed in Part A of the experiment. The student dissolves all of the brass sample by reacting it with 7 M nitric acid. (Clearly, the dangers of using 7 M nitric acid and 2 M ammonia (vide supra) need to be articculated (with only one ‘c’) to the student before they start the experiment.) The copper in the brass is oxidized to the copper(II) aqua ions in solution. This gives the solutions a light blue tint. The absorbance of light by the copper(II) ion in this solution is characteristic but not particularly intense. It is necessary to increase the absorptivity of the copper species to more fully utilize the dynamic range of the Spectronic 20, so a stronger-field ligand (ammonia) is added to the solution to work as a spectroscopic amplifier. The original copper(II) solution is added to a volumetric flask and 2.0 M ammonia solution is added to the mark. The copper(II) reacts with the ammonia to make a Cu(II)-NH_3 complex (Cu^{2+} for simplicity) that has a deep blue color [8].

2. In Part B, the students collect data for a calibration curve. When the students plot the absorbance of light by Cu^{2+} at 550 nm versus the concentration of their solution, they obtain a linear relationship. This is due to the familiar Beer’s Law [9] (equation 1):

$$A = \varepsilon bc \tag{1}$$

where A is absorbance, ε is the molar absorptivity of the complex, b is the path length and c is the concentration of the solution. With this relationship, they can determine the concentration of any similar Cu^{2+} solution that absorbs an amount of light corresponding to the absorbance values between the most and least concentrated solutions.

3. With the calibration curve in hand, they can measure the effect of 550 nm light on their unknown sample and calculate the concentration of the Cu^{2+}. Having calculated

the concentration and knowing both the total volume of the solution and the mass of copper in the original sample, the percent copper in the solid sample can be calculated.

Why This Experiment?

As a test experiment for setting up CGI scripts for student interaction and posting of data, "*Analysis of Brass*" has some advantages.

1. The laboratory was developed in-house so we need no permission to use it.

2. All analyses are numerical. There are no subjective observations like "deep blue" that can be described in a myriad of ways.

3. This laboratory allows us to learn how to deal with numerical input. There are many ways a student might type in a number (e.g., 0.00400, 0.00400, 4.00e-3, 40.0E-4, etc.), but they can be accommodated by incorporating regular expressions into the scripts. A regular expression is "a pattern that you can use to find what you're looking for when it varies from case to case" [10].

4. There are only simple calculations and analyses involved, such as transforming percent transmittance to absorbance and linear regression.

5. It is a two part analysis. We could learn to set up the scripts so that the student can finish one part, then return and finish the other. If we can do this for a two part analysis, we can adapt and expand as necessary for other more complicated experiments.

6. Many scripts, for example one used to find the student's name and section, can be used in any subsequent set of scripts.

The Series of Scripts

Copies of the scripts are available in the supporting material (36jl1897.zip). The script names have the suffix "_exp7a" which is the designator for "Analysis of Brass" in our current *General Chemistry Laboratory Manual*. This suffix has been removed from the titles of the following sections for clarity. A flow chart illustrating the movement of the student through and within the series of scripts is given in Figure 1.

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

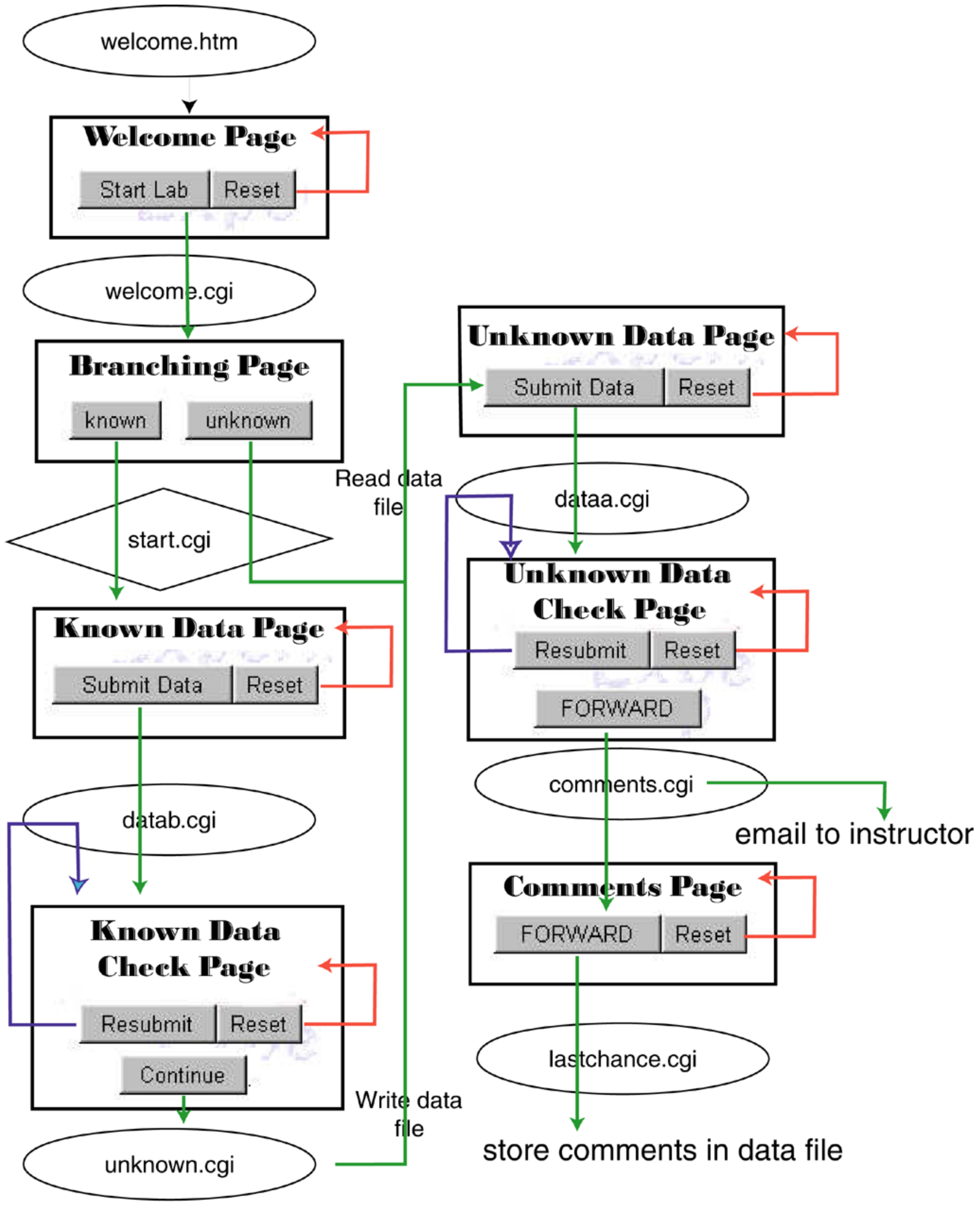


FIGURE 1. FLOW DIAGRAM OF THE OPERATION OF THE SERIES OF CGI / PERL SCRIPTS FOR THE COLORIMETRIC EXPERIMENT, “ANALYSIS OF BRASS.” RED LINES REFER TO RESET FORM ACTIONS, WHICH CLEAR AND RELOAD THE CURRENT PAGE WITHOUT RERUNNING THE CGI SCRIPT. BLUE LINES REFER TO FORM ACTIONS WHICH LAUNCH THE SAME CGI SCRIPTS WITH NEW INPUT DATA. GREEN LINES REFER TO FORM ACTIONS WHICH LAUNCH NEW CGI SCRIPTS AND CREATE NEW PAGES.

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

welcome.htm

Welcome.htm is the first page of the laboratory that the student encounters (Figure 2). The file is a standard html (hypertext markup language) page. All of the control elements, such as font, boldface, headings, forms and links, are given as ASCII text. It is the job of the web browser to interpret the elements and text and make the page. Although we used a commercial product to construct html pages [11], many word processors can save text files in html format.

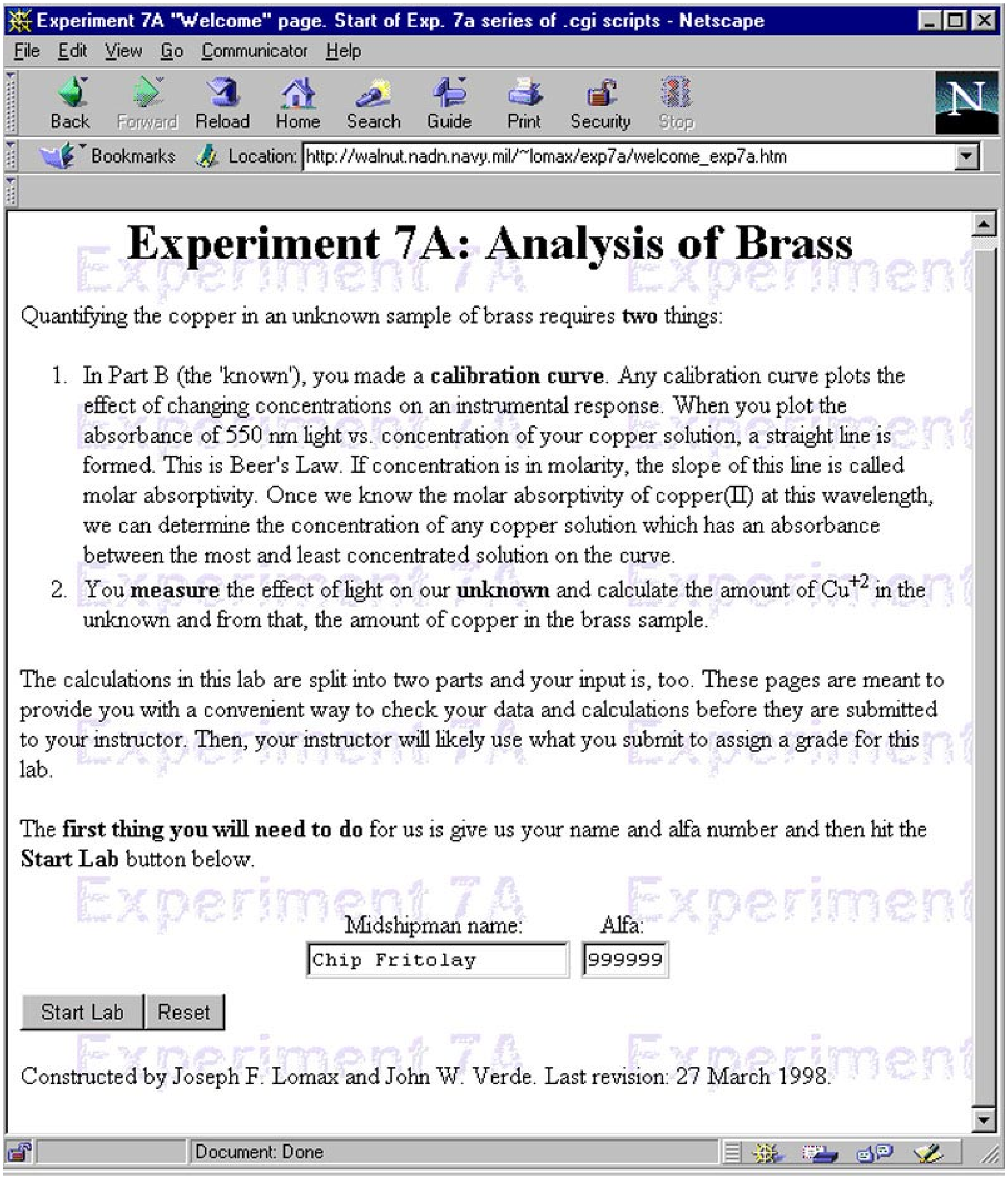
Welcome.htm gives a very brief introduction to the laboratory and relates what the student needs to complete before he or she starts with the remaining pages (perform the laboratory and do the calculations). At the bottom of welcome.htm is a web form element. A form element is an enclosed group of web elements. These elements can be used to generate a place where the student can enter information, in this case their name and student identification number, an “alfa number” in naval-academy-speak. If what they enter is not satisfactory, they can clear it by selecting the “Reset” button. Each page has a “Reset” button that causes the current page to be recalled without any input data; however, if the student is satisfied, he or she can continue by selecting “Start Lab.”

welcome.cgi and lab.pl

By selecting “Start Lab,” the student has instructed the form to start a series of actions. In particular, it calls into action the program welcome.cgi. This program resides on the server; it is not called onto the web browser. Through a Common Gateway Interface or “CGI,” welcome.cgi allows information to be input into the server and manipulated and it creates a new webpage (and within it another form). CGI is a protocol that allows information to be passed from the user to the server to the program and back, and the extension, .cgi, tells the browser that a program using this protocol is on the way.

Even though the programs may end up being quite long, in the vernacular they are called CGI scripts. The programming language that we used in all these CGI scripts is Perl 5.0 (practical extraction and report language) [12]. As programs go, welcome.cgi is relatively simple. After pulling in the data from the welcome.htm form, the script calls a subroutine, &find_mid_name (the ‘&’ designates a subroutine in Perl). The subroutine, &find_mid_name**,** “asks” a number of questions. Is the alfa number valid?

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8


Experiment 7A "Welcome" page. Start of Exp. 7a series of .cgi scripts - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://walnut.nadn.navy.mil/~lomax/exp7a/welcome_exp7a.htm
Document: Done

Experiment 7A: Analysis of Brass

Quantifying the copper in an unknown sample of brass requires **two** things:

1. In Part B (the 'known'), you made a **calibration curve**. Any calibration curve plots the effect of changing concentrations on an instrumental response. When you plot the absorbance of 550 nm light vs. concentration of your copper solution, a straight line is formed. This is Beer's Law. If concentration is in molarity, the slope of this line is called molar absorptivity. Once we know the molar absorptivity of copper(II) at this wavelength, we can determine the concentration of any copper solution which has an absorbance between the most and least concentrated solution on the curve.
2. You **measure** the effect of light on our **unknown** and calculate the amount of Cu^{+2} in the unknown and from that, the amount of copper in the brass sample.

The calculations in this lab are split into two parts and your input is, too. These pages are meant to provide you with a convenient way to check your data and calculations before they are submitted to your instructor. Then, your instructor will likely use what you submit to assign a grade for this lab.

The **first thing you will need to do** for us is give us your name and alfa number and then hit the **Start Lab** button below.

Midshipman name: Chip Fritolay
Alfa: 999999

Start Lab | Reset

Constructed by Joseph F. Lomax and John W. Verde. Last revision: 27 March 1998.

FIGURE 2. SCREEN CAPTURE OF THE OPENING PAGE OF THE SERIES OF SCRIPTS, WELCOME.HTM.

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

That is, is it a number and not text? Is it the correct length? Does it correspond to a midshipman, commonly referred to as a “mid,” currently taking the course? The script then extracts the name of the mid from a file on the server. Finally, the script sends a new webpage to the mid and includes information derived from find_mid_name, such as the typed-in name, the extracted name and the alfa number (if it is really valid, otherwise it reports an error). To keep the subroutine portable, the file name is listed as the scalar variable, $namedat_url. (A scalar is a simple value such as a number or a string of text. In Perl, it is denoted by a leading $.) The text value of $namedat_url can be changed in the CGI script by the programmer to whatever path and name the student data file may have.

&find_mid_name is an example of a subroutine that is needed for this and other web pages. Whenever we created such a subroutine, we placed it in a file called the script library, which we named, lab.pl. Each script that used subroutines from lab.pl would need to have this line:

```
require “/usr/path_to_file/lab.pl”;
```

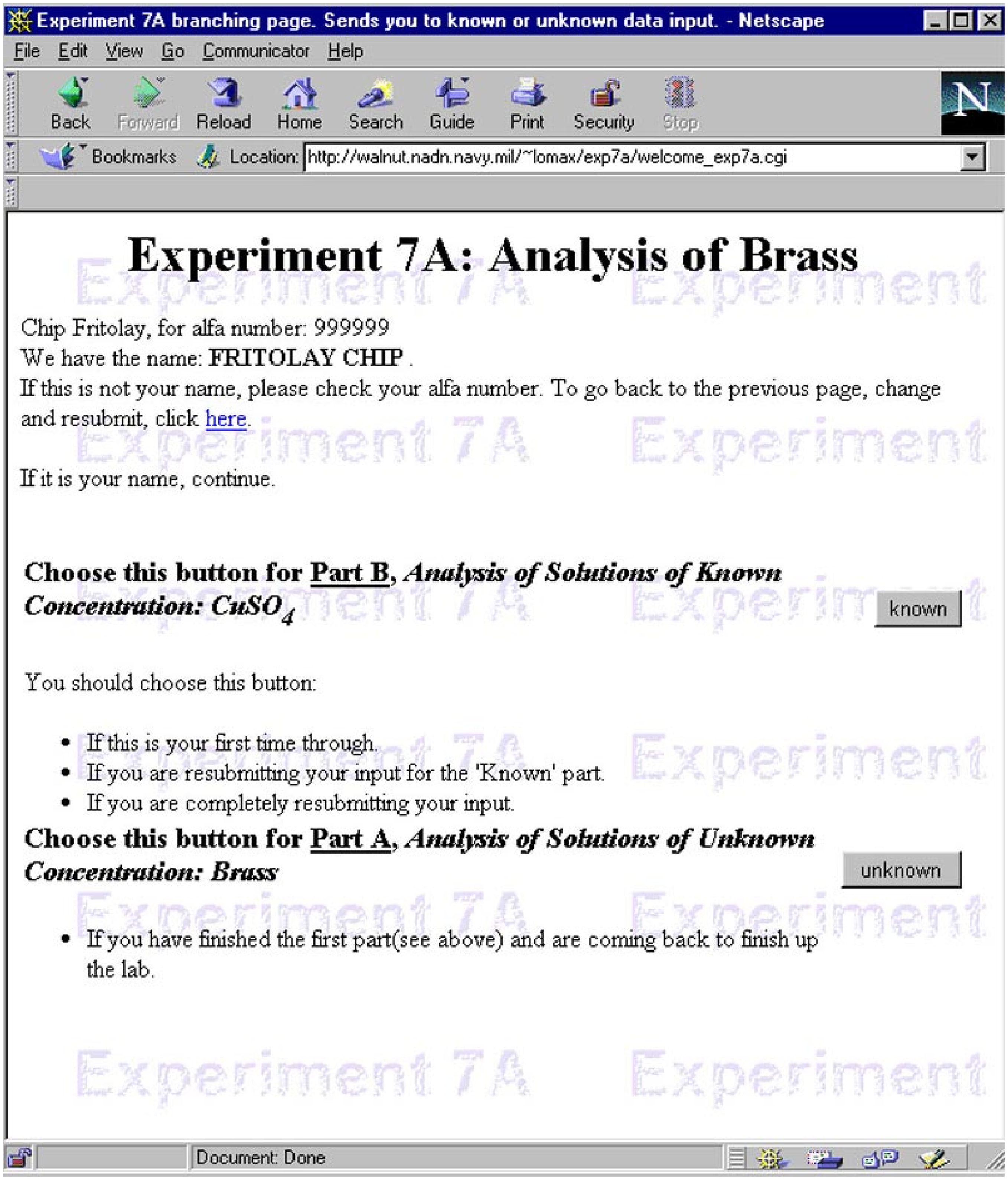
This essentially makes the library part of the script, so that it can find all the necessary subroutines. The other script library we used, which has quite a number of useful CGI scripting subroutines, is cgi-lib.pl [13]. It, too, needs a “require” statement.

The new page created by welcome.cgi allows the student to check that the name extracted from the alfa information is correct, but its major function is as a branching page (Figure 3). It allows the student to go to the calibration curve portion of the analysis or if that has been finished, go on to the analysis of the brass sample. By selecting either the “known” or “unknown” buttons, a specific subroutine that sends the mid to a specific new html page is run.

start.cgi

By choosing the “known” path, you will proceed with the calibration curve analysis starting with start.cgi. The html page formed by following this path, again, has text and a form. In the text the students are warned to have the correct number of significant figures in their input values. The text tells the student that this is the data input section and gives advice on how to copy data directly from a spreadsheet and paste it into the table. Our students are required to purchase a personal computer, and, for the last decade, all of these computers have had a spreadsheet loaded, currently Quattro Pro. In

THE CHEMICAL EDUCATOR
© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8


Experiment 7A branching page. Sends you to known or unknown data input. - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://walnut.nadn.navy.mil/~lomax/exp7a/welcome_exp7a.cgi
known
unknown
Document: Done

Experiment 7A: Analysis of Brass

Chip Fritolay, for alfa number: 999999
We have the name: **FRITOLAY CHIP** .
If this is not your name, please check your alfa number. To go back to the previous page, change and resubmit, click here.

If it is your name, continue.

Choose this button for Part B, *Analysis of Solutions of Known Concentration: $CuSO_4$*

You should choose this button:

- If this is your first time through.
- If you are resubmitting your input for the 'Known' part.
- If you are completely resubmitting your input.

Choose this button for Part A, *Analysis of Solutions of Unknown Concentration: Brass*

- If you have finished the first part(see above) and are coming back to finish up the lab.

FIGURE 3. SCREEN CAPTURE OF THE BRANCHING PAGE, WELCOME.CGI.

THE CHEMICAL EDUCATOR
© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

the current school year the students also have been required to purchase a graphing calculator, the TI-92. About half used the spreadsheet and half used a calculator (See Table 1, Question 5). After the data is input into the table, the next CGI script, datab.cgi, is launched by selecting “Submit Data.”

datab.cgi
As before, once students have submitted the data, a response html page is generated. There are a few differences between the html page formed by datab.cgi and the previous one made by start.cgi. This page redisplays their input and may display errors, indicated by a Charlie flag for a calculation error as shown on the fifth concentration, or both a Sierra and a Foxtrot flag for incorrect significant figures as shown on the second concentration and the fourth % Transmittance (Figure 4).

How did these flags appear (or not)? This is the central part of the function of the datab.cgi script. It also points to where programming begins to assist pedagogy.

The major reason for using CGI scripts in this laboratory is to give the students rapid feedback on the calculations and analysis reported. Sequentially, the CGI script, datab.cgi, (1) pulls in the input data; (2) checks to see if all data are numerical; (3) calculates absorbances from input percent transmittances; (4) checks to see if the solution concentrations are correct; (5) takes the absorbance calculated by the script and the solution concentrations and runs a linear regression subroutine to calculate the slope and intercept of the calibration curve; (6) checks to see if the concentration, percent transmittance data, and regression analysis submitted by the student have the correct number of significant figures (three in these cases); (7) checks the regression slope and intercept against the absorptivity and intercept the student has input; (8) creates a data file tagged to the alfa number; and (9) creates a new html page.

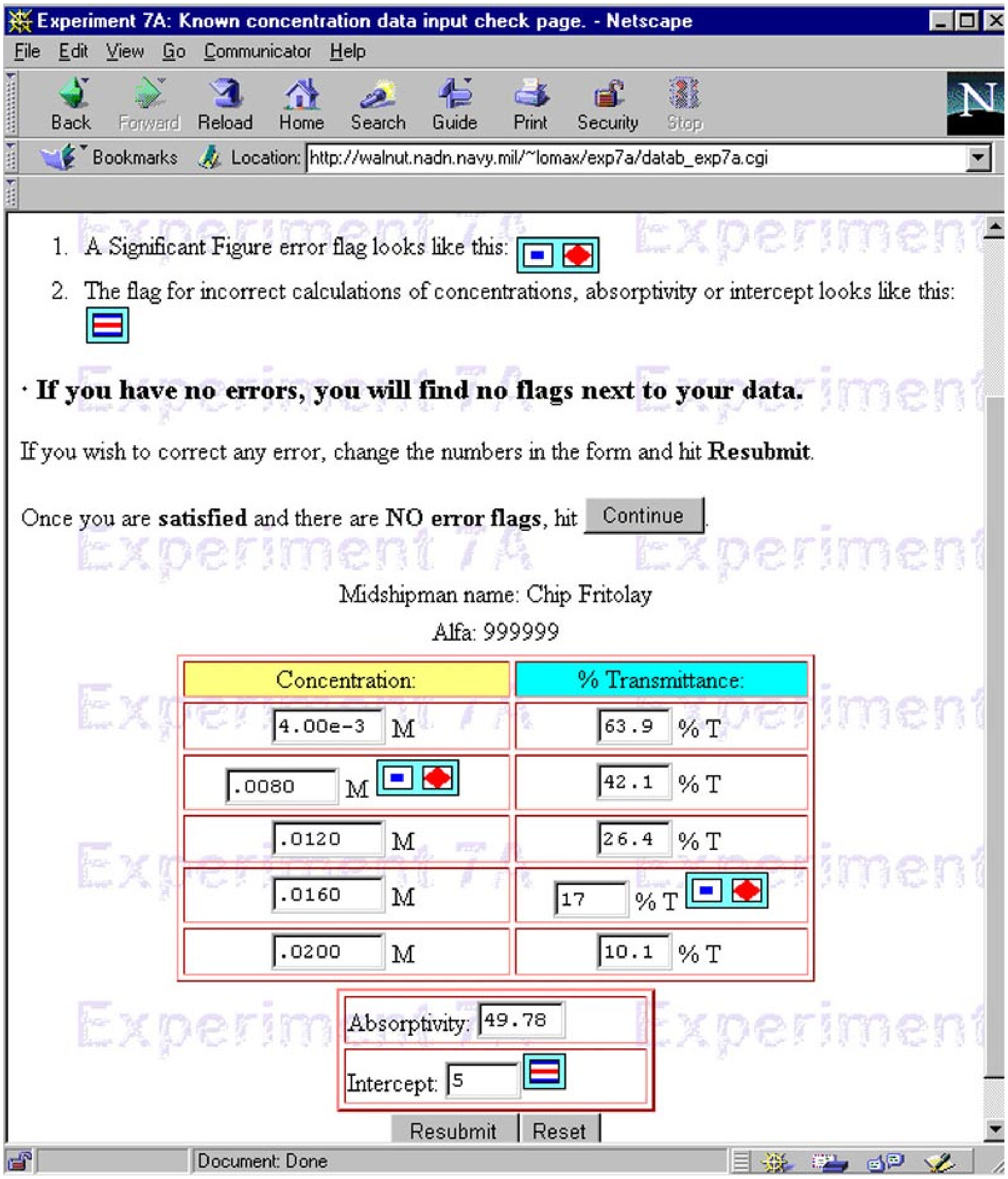
The new html page has their data and the appropriate flags for significant figure (Sierra-Foxtrot) and calculation (Charlie) errors. If no data or analysis is flagged, the student may select the “Continue” button, and this will launch the next CGI script, unknown.cgi; however, if the student finds error flags he or she will know immediately that something is wrong. The international signal flags are used because they clearly indicate where the error is. These flags remain until the student corrects the error, so their presence is persistent but not insulting. Hopefully, the student will find the error, correct it, and then select the “Resubmit” button. This sends the revised data through

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

TABLE 1. Responses from Radio Button Comments. SC151 is the one semester General Chemistry course for students who have validated one semester. SC111 is the first of two semesters of General Chemistry. These particular students either are repeating SC111 or they came from a remedial course. Not all students responded to all questions, except the first. The only default answer on any question was the “No Comment” on the first question.

	SC151 (n = 77)	SC111 (n = 99)
1. How well did these webpages help you understand the calculations?		
They did not harm me.	21	28
They helped me a little.	16	27
They help me quite a bit.	32	35
They were confusing instead of helpful.	5	7
No Comment.	3	16
2. Would you like to see this method used for other laboratories?		
Yes	69	84
No	5	13
3. How many times did you go through this set of web pages?		
One	35	38
Two	16	19
Three	13	19
Four	2	5
More than Four	9	16
4. In this final (or only) time, how long did it take you to make it through the webpages from when you started them until now?		
Less than five minutes	24	26
Five to eight minutes	14	17
Eight to fifteen minutes	17	24
Fifteen to thirty minutes	8	11
More than thirty minutes	13	21
5. What type of calculation device did you use to analyze the data?		
Quattro Pro (a spreadsheet)	34	49
TI-92 (a required calculator)	32	39
Other calculator	5	6

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8


Experiment 7A: Known concentration data input check page. - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://walnut.nadn.navy.mil/~lomax/exp7a/datab_exp7a.cgi
Document: Done

1. A Significant Figure error flag looks like this:
2. The flag for incorrect calculations of concentrations, absorptivity or intercept looks like this:

- **If you have no errors, you will find no flags next to your data.**

If you wish to correct any error, change the numbers in the form and hit **Resubmit**.

Once you are **satisfied** and there are **NO error flags**, hit Continue.

Midshipman name: Chip Fritolay

Alfa: 999999

Concentration:	% Transmittance:
4.00e-3 M	63.9 % T
.0080 M	42.1 % T
.0120 M	26.4 % T
.0160 M	17 % T
.0200 M	10.1 % T

Absorptivity:	49.78
Intercept:	5

Resubmit Reset

FIGURE 4. SCREEN CAPTURE FOR THE DATA INPUT PAGE, DATAB.CGI. NOTE THAT THERE ARE A NUMBER OF ERRORS FLAGGED BY INTERNATIONAL SIGNAL FLAGS. THE “S” AND “F” (SIERRA AND FOXTROT) ARE FOR SIGNIFICANT FIGURE ERRORS AND THE “C” (CHARLIE) IS FOR A CALCULATION ERROR.

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

the datab.cgi script and the same calculation and analysis checks described above and recreates the html, hopefully with fewer or no error flags. Once the student is satisfied, they can select “Continue.” This page is designed so that the student can continue with or without errors in the analysis.

unknown.cgi
The text at the top of this page, created by unknown.cgi, includes data (name, alfa, and graphical slope and intercept) that has been brought along from the previous page. The form within the page has slots for further data and analysis input. The data is the unknown brass sample mass and the percent transmittance of the unknown solution, and the analysis is the concentration determined from the calibration curve as well as the percent copper by mass. There are intermediate steps involved in the analysis, but we felt that these were sufficient to allow the student to find the source of any mistakes that were flagged. As before, the student is expected to input the data and analysis and select “Submit Data.” This page can also be accessed from the program start.cgi if the student selects “unknown” on the welcome.cgi page; however, if the student chooses “unknown” on the welcome.cgi page without having previously completed the “known” portion (and had the data and analysis resident in an alfa tagged data file), an error page is returned.

dataa.cgi
Selecting “Submit Data” on the unknown.cgi page causes the form to launch the script dataa.cgi. It may seem curious to the reader that datab.cgi comes before dataa.cgi. The “b” and “a” refer to the sections in the laboratory instructions. The decomposition of the brass is the first action started in the laboratory (Part A), and subsequently, the unknown solution is made by diluting the resulting $Cu(NO_3)_2$ solution with the 2.0 M ammonia solution. Part B of the laboratory includes making the standard solutions and the calibration curve. The standard solutions are made as the decomposition of the brass proceeds in a fume hood (NO_2 gas is evolved). The names of the scripts reflect an idiosyncrasy in the long established experiment.

Many of the same data manipulations and error flag placements occur in dataa.cgi as occurred in datab.cgi, except that simple algebraic calculations involved in the “unknown” calculations have replaced those associated with the linear regression in the calibration curve. In addition, the data file has been retrieved, its data used, and additional data has been added to the file. To finish off the laboratory when the student is satisfied, “FORWARD” is selected.

comments.cgi
Transparent to the student, selecting “FORWARD” activates comments.cgi. This script sends off the laboratory analysis to the instructor. In this way the instructor still gets the analysis from the student even if the student does not desire to fill out the comment page. In the usual large general chemistry course, there is only one instructor lecturing to many students; so, mailing the students’ grades to the instructor involves only one email address. At the Naval Academy, however, general chemistry has approximately 25 instructors teaching 50 sections of 20 students each. All sections do the same experiments. In order to send the analysis to the proper instructor, the student’s section is retrieved using &find_mid_section, and this section number is used in &find_instructor to retrieve the instructor’s email address from the userno.dat file. Both subroutines are found in lab.pl, and a “require” statement is placed in comments.cgi for the file userno.dat. The data and analysis from student input and the computed analysis are sent to the instructor by email using a formatted MAIL routine.

The html page formed by comments.cgi allows the student the ability to send comments about the series of web pages. Within the form are two input styles: “radio button” and text. Each time a radio button comment is chosen an associated array is called, and the comment phrase connected to that radio button is recognized as an array key. The value associated with this comment phrase key is incremented by one. This is how we obtained the count on the comments summarized in Table 1 and discussed in the “Pedagogical Considerations” section. The written data comments are appended to files. When the student is satisfied with the comments given, he or she selects FORWARD.

lastchance.cgi
The final script, lastchance.cgi, informs the student that the series of web pages is finished, and that the analysis has been sent off to the instructor. Nothing is in place to stop the student from rerunning the whole set of scripts. The result would be another email message to the instructor and it would be up to the instructor to accept the latest analysis.

Error Pages

In our set of scripts the student will get an error page if he or she (1) does not put in an alfa number in welcome.htm or if there are any nonnumerics (or having input a valid but an incorrect alfa into welcome.htm), (2) tries to go to the unknown in start.cgi without having gone through the known part previously, (3) tries to start from the middle of the series of pages, (4) tries to put in text or a zero value for absorptivity (as text is given a numerical value of zero in Perl, any text will give this, too). The first three will cause fatal errors if not caught because the necessary data file is tagged to the alfa number. The final error has to be trapped because a zero value for absorptivity will cause a division by zero in a subroutine and propagate a fatal error.

Pedagogical Considerations

The reasons for using CGI scripts for reporting laboratory data are: (1) to give the student rapid feedback and (2) to allow them to report their data quickly. These are not online tutorials and are not meant to be. Students encountering significant difficulties are given a message that encourages them to seek assistance from their instructor. An added benefit to the instructor is that the students' results can be reported in a form that allows for quick evaluation of their efforts.

Each data page allows the student to input data and receive feedback on the accuracy of calculations as well as whether or not the proper number of significant figures were used. If the student's calculations and analysis are all correct, he or she can make it through this set of web pages rapidly. About half of the students were able to complete the set the first time through (Table 1, Question 3), and to do so quickly (Table 1, Question 4).

Although the rapid positive feedback and reporting benefited those students who had all of the calculation and analysis correct, the real benefit is for those who were unaware they had errors. Although hints or more human responses could have been programmed in, the neutral advisory flag was chosen. This de-emphasizes program and programmer and emphasizes the student's responsibility for finding the error and correcting it.

In the initial set of comments, it was found that the number of students taking four or more times through the page corresponded well with the number who took more than

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

30 minutes to complete the exercise. It was clear that those with the greatest difficulties were not benefiting from the pages; they were spending too much time, getting frustrated, and blaming the method. To deal with this, in each data page, a counting subroutine, &data_counter, was created so that it could be called from the lab.pl file. Now, each time the student calls up a particular data page, the counter in &data_counter autoincrements by 1. If the student is not finished by the fourth time through a particular data page, a page like the one in Figure 5 comes up. Note that the responsibility for getting help is still given to the student. If the student chooses to seek help and if the problem is rooted in the student's inability, the instructor finds out quickly and can help. If, instead, the problem is related to programming, the instructor can contact the webmaster and a fix can be implemented before too many students get frustrated.

Generally, the program was exceptionally well received (Table 1, Question 1) and, in particular, the students were interested in seeing more laboratory reports done in this manner (Table 1, Question 2). It is the hope of the authors to accommodate them with further laboratory reports using CGI scripts in the near future.

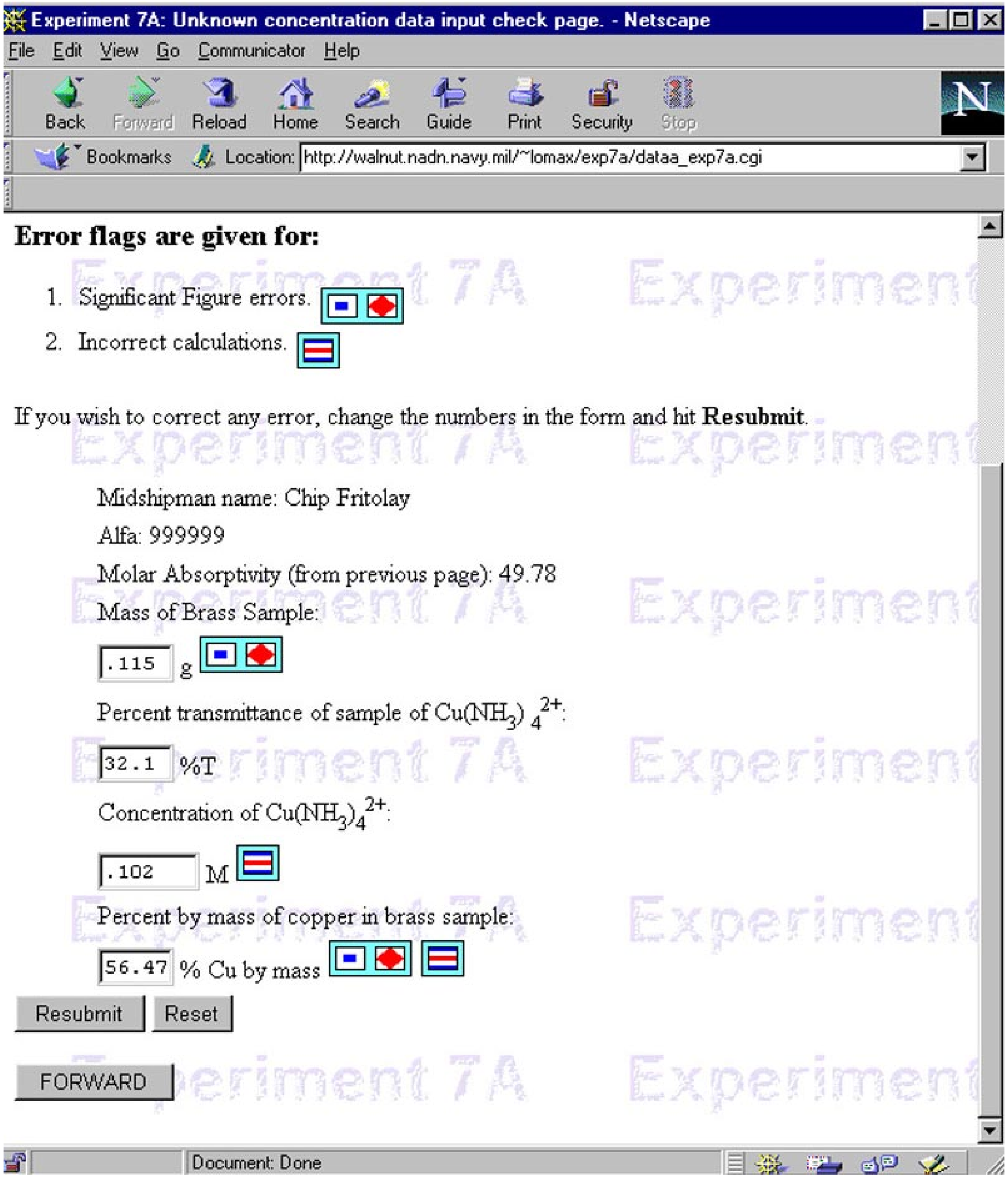
Programming Considerations

The purpose of this paper is to show the method and rationale behind our work so that it can be adapted in schools that have large laboratory sections and web access. The purpose is not to provide a specific laboratory for the general chemistry teaching community to use, nor is it to give them a set of CGI scripts to use, though we hope that both will happen. The following programming considerations are given for the chemist who is not a programmer, but wants to provide the student in large sections with a better laboratory experience.

CGI Scripts

To the uninitiated chemist, using CGI scripts can seem foreign; however, they can be used to efficiently transfer information across the web. In our environment, where all students have web access and fast data transfer, we find this type of program, which is resident on the server, to be more efficient and easier to create and modify than programs that are sent to the student's computer, such as Java Scripts. For us, this made learning how to write GGI scripts worthwhile. Our intent for giving these programming

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8


Experiment 7A: Unknown concentration data input check page. - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://walnut.nadn.navy.mil/~lomax/exp7a/dataa_exp7a.cgi
Document: Done

Error flags are given for:

1. Significant Figure errors.
2. Incorrect calculations.

If you wish to correct any error, change the numbers in the form and hit **Resubmit**.

Midshipman name: Chip Fritolay

Alfa: 999999

Molar Absorptivity (from previous page): 49.78

Mass of Brass Sample:

.115 g

Percent transmittance of sample of $Cu(NH_3)_4^{2+}$:

32.1 %T

Concentration of $Cu(NH_3)_4^{2+}$:

.102 M

Percent by mass of copper in brass sample:

56.47 % Cu by mass

Resubmit Reset

FORWARD

FIGURE 5. A SCREEN CAPTURE OF A PORTION OF THE DATA INPUT PAGE DATAA.CGI WITH THE WARNING NOTE.

considerations is so that others can adopt and adapt our scripts to their particular needs and experiments. Of course, the more you know about the programming language and its common usages, the larger the variety of adaptations you can make.

The language used for most CGI scripts is Perl. As programming languages go, Perl is easy to learn; however, an appreciation of programming logic, loops and/or regular expressions from other languages will certainly speed the transition. There are dozens of books on CGI/Perl, but the most popular book for learning Perl is *Learning Perl* [14], and the reference and more advanced technique text for the field is *Programming Perl* [10], also known as “The Camel Book” because of the illustration on the cover. They contain explanations on how to download and install Perl on your system whether you use a Unix, Windows, or Mac operating system for your CGI server. Your choice of which book you want to use for learning CGI scripting is mostly determined by your level of programming proficiency: introductory, “cookbook,” accomplished, or expert [13, 15, 16]. You might want to choose a book that includes a CD-ROM with a Perl compiler, a number of script libraries, and example scripts that cover some of the functions that you might want to accomplish.

Truth be told, the CGI script can be written in any language that the server can compile, but there are certain advantages to using Perl besides its ease of learning. There are many Perl/CGI scripts available on the web, such as those found in Matt’s Script Archive [17]. Perl scripts tend to be portable; that is, they can be loaded onto many systems. Thus, often you can piece together the programming functions from available sources.

Properly assembling the components of a set of scripts involves an exceptional amount of detail work. A paper in this Journal by Earp and Tissue [1] describes the general operation of CGI scripts. In the following sections, we present some other considerations that you need to be aware of if you intend to implement either this set of web pages or similar sets.

Student Identification Numbers and Filenames

Data is carried through posting of scripts by two methods: hidden variables in forms and writing/retrieving files. Hidden variables are objects within a form element that

ISSN 1430-4171
THE CHEMICAL EDUCATOR
http://journals.springer-ny.com/chedr
© 1998 SPRINGER-VERLAG NEW YORK, INC.
S 1430-4171 (98) 06258-8

allow data to be brought forward to the next script without the student re-inputting the data. A command would look like:

```
<INPUT TYPE="HIDDEN" NAME="alfa" value="$alfa">
```

The information within the < > brackets is a command, specifically an INPUT command whose type is HIDDEN. The data for this hidden variable originated from student input and carries along the most important hidden variable in these scripts, the alfa number. Any time a CGI script is called and no alfa number is brought forward (for example, if the student returns to a page using a bookmark), an error page is produced with a link to the first page. Alfa numbers are used because they are unambiguous. You do not need to worry about two students having the same name, such as David Robinson, or the student using a name that does not exactly fit the version in the data file. The latter is a particular concern when the student name field in the data file is too short for the student's whole name.

The scalar variable, *$alfa*, is used in separate subroutines to find the student name, section, and instructor. If this set of scripts were set up at a different school, a data file with data for each student on each line for each course would be necessary. The subtle differences in the data file (e.g., student number length, name field length, course name field) would have to be accommodated in the regular expressions used to find and retrieve information from this file.

Numerical data, such as the absorptivity, can also be carried to later forms. For it to be presented in the form as a nonempty scalar variable, a line needs to be put into the script that reads, for example,

```
$Absorptivity = $input{Absorptivity};
```

This puts the value associated with the name "Absorptivity" that is coming from the input array, into the scalar variable, *$Absorptivity*.

As mentioned above, the other manner with which one can carry along data is by opening a file, putting data into it, and retrieving it. An example of a subroutine that opens a data file and puts data into it is

```
&open_mid_data($alfa);
```

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

In the subroutine, &open_mid_data, the alfa number is carried in and used to name the file in the statement:

```
open(MIDDATA,”>$def_data_dir/”.”$alfa\.dat”) || die $!;
```

The filehandle (a temporary name for the file until it is defined later), MIDDATA, is used to put data into (the > means to append to a file) the file named by its default data directory and alfa number, such as /usr/path_to_file/data/999999.dat. If the file is unable to be opened, the script dies and an error is sent to the standard output (*$!*). In addition to making data available at any point in the set of scripts, the data file is used in two more ways. If the student stops after the known input, he or she can start again at the welcome.htm page, input their alfa at the branching page, then go directly to the unknown page. The alfa number is also used as a key to student information to begin the process of sending analysis and data to the instructor. From the course data file the alfa number keys the section number. In another file, the section number keys the instructor’s email address. Pertinent data and analysis are sent to the instructor with the name and alfa number of the student.

Numbers and their Manipulation

Any time numerical data is input, it is checked to make sure it contains neither letters (other than “e” for a base-ten exponent) nor more than one decimal point. If either of these is found, it is flagged with “not a #” in the data slot of the newly posted web page; however, a separate check subroutine is put in when nonnumeric data in a numeric field results in the program performing a division by zero (Perl defines the numerical value of text as zero). If this check finds an improper zero, it preempts the division by producing an error page with an explanation. Without such a planned interruption, one gets the infamous “Error 500.” This sight is the one guaranteed source of dissatisfaction from the student as it stops the process, but gives no information on the source of the error.

It is always better to try out any conceivable error, such as putting in text into a numeric field, find out what occurs, and accommodate it. Often this can be accomplished by making a more sophisticated regular expression or looping routine; however, when reasonable efforts (and you need to decide what they are) do not accomplish a fix, have an error page with an explanation and link to the proper repair.

© 1998 SPRINGER-VERLAG NEW YORK, INC.
ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

Certainly, there will be errors that students will discover that were inconceivable to you. We have found the data-throughput counter and the comments page useful additions to these pages that allow us to pick up programming bugs and correct them more quickly.

The programming in the calculations presented the least difficulty of all the programming. The first program written was a least-squares fitting program [18]. Certainly, there are more efficient programs to compute regressions, but it was a good test of our ability to create and use loops within subroutines for a program. This program took student input of concentration and percent transmittance and gave values for slope and intercept against which the corresponding student input was checked for validity. Note that &regression calculates the molar absorptivity from the student's input concentration data and computer calculated absorbance.

One of our most important programming considerations turned out to be the method for determining the validity of the student's input. The correctness of an answer can be checked by requiring the entered number either be (1) within a certain fraction of the computer calculated value or (2) within a set range. This was more of a pedagogical difficulty rather than a programming one. We chose to measure the accuracy of the reported molar absorptivity by determining whether or not it was within ±3% of the slope calculated by the &regression subroutine.

One example of why this decision is critical is in checking the value for the *y*-intercept from the linear regression. In a colorimetric experiment, the *y*-intercept should be zero. We found that being within a fraction of the calculated value was not a valid measure of the accuracy. Consider this scenario, which has happened to us a number of times. A student takes their percent transmission data, calculates absorbance, and enters this data with three significant figures into their calculator's regression function. The intercept value they get is 0.001 absorbance units; however, the CGI script calculates absorbance as part of its routine and carries 15 significant figures into the regression program and gives an intercept value of – 0.001. These two values are within 0.002 of each other and should not cause an error, but – 0.001 is 200% away from the true value of +0.001 absorbance units. Returning an error message when this occurs irritates the students and detracts from their seeing this method of evaluation as a positive tool.

Normal and Associative Arrays

An array is an ordered list of scalar data. The utility of arrays is that a set of related values (numerical or text) can be saved in an organized manner without needing to have a independent name for each variable. An example where a normal array (designated by an initial @) is used is the recording and checking of the values for concentration. The array for our list of concentrations could look like: @conc = (0, 4e-3, 8e-3, 1.2e-2, 1.6e-2, 2e-2).

The last five values in parentheses are values that the students entered into the table. For ease of use in our subroutines, the initial value in the array is $conc[0] = 0. Note that (1) the title of this array scalar, conc, is the same as the title of the array, but is preceded by the scalar marker, $; (2) the key to the scalar value, the 0 in hard brackets, is always an integer; and (3) the initial key to a normal array is 0. As the data comes into the subroutine, &check_conc, we put them into the array, one at a time, and when we need them, we take them out one at a time.

For an associative array (also known as a hash, named by the shorthand for the symbol %), the key can be any number or text. In other programming languages what Perl calls a key may be called a pointer. You can input the values for a mathematical function using a hash, with the *x* values being the keys for the *y* values. In our scripts, we most commonly use hashes for writing and retrieving data into a file. Each piece of data is given a variable and a value separated by a colon, such as Absorptivity: 30.3. In this way the key (Absorptivity) and the value (30.3) are associated, but can be taken out of the file, put into a hash, and used together or separately as needed.

Reporting Data and Analysis to the Instructor

We chose to report the data and analysis to the instructor by email because of the ease of setting up a formatted MAIL routine. Perl allows you to format a mail message using variables: scalar, keyed, or results from subroutines. In this way, you can get a standard format with each student's input analysis and the calculated analysis using the student's data. There are better ways of conveying the student's report to the instructor than emailing him or her the results. For example, a file ordered by student section and name can be used. This could be opened, appended, ordered, and closed each time it is called. Creating such a script is a bit of a programming challenge, but as it did not have a direct impact on the pedagogy, we felt it could be left until later. That being said, it is the next major improvement that will be made on the set of scripts.

Conclusion

The series of CGI scripts for the experiment “Analysis of Brass” provide the student with prompt, useful feedback on calculations and analyses. Most students who worked through the scripts within three tries reported that they liked the method and the instructors were able to evaluate the students rapidly and precisely. Anecdotal evidence suggests that the students performed better using the scripts, but no control data was available.

The most overwhelming measure of the student’s acceptance of the use of the scripts was given by their expressed desire that they would like to see this method used for other laboratories (Table 1, Question 2). We hope to accommodate them in the near future. In addition, the Chemistry Department at the Naval Academy is moving towards electronic publishing of our general chemistry laboratory manual. With electronic publishing revisions, additions, subtractions, and adaptations can be rapidly incorporated and we can increase the quality of the graphics. It is anticipated that the electronically published *General Chemistry Laboratory Manual* and web-based analysis scripts will be integrated as more CGI script series become available.

ACKNOWLEDGMENT

We would like to thank Steven J. Stuart, a proper Perl hacker, for his ability to break our programs and his helpful suggestions on how to fix them. We would like to thank Edward Koubek for writing the original laboratory “Analysis of Brass.” We would like to thank John Schultz for the inspiration of the Wiz, a series of mainframe Basic programs. This work was funded by U.S. Naval Academy Curriculum Development Program and the Science and Education Apprenticeship Program (DoD).

REFERENCES

1. Earp, R. L.; Tissue, B. M. *Chem. Educator* **1996**, *1(5)*:S 1430-4171(96)05055-8. Avail. URL: http://journals.springer-ny.com/chedr/.

2. Lipkowitz, K. B. *J. Chem. Educ.* **1989**, *66*, 275.

ISSN 1430-4171
http://journals.springer-ny.com/chedr
S 1430-4171 (98) 06258-8

3. DeKock, R. L.; Madura, J. D.; Rioux, F. C.; Joseph, R. L. In *Reviews in Computational Chemistry*; Lipkowitz, K. B.; Boyd, D. B.; Eds.; VCH Publishers: New York, 1993; Vol. 4, pp 149–228.

4. Fitzgerald, J. P. *J. Chem. Educ.* **1993**, *70*, 988.

5. Lipkowitz, K. B. *J. Chem. Educ.* **1995**, *72*, 1070.

6. Lipkowitz, K. B.; Pearl, G. M.; Robertson, D. H.; Schultz, F. A. *J. Chem. Educ.* **1996**, *73*, 105.

7. Zielinski, T. J.; Allendoerfer, R. D. *J. Chem. Educ.* **1997**, *74*, 1001.

8. Cotton, F. A.; Wilkinson, G. *Advanced Inorganic Chemistry*, 5 ed.; Wiley Interscience: New York, 1988.

9. Beer, A. *Gundriss des Photometrischen Calcüles*; Friedrich Vieweg und Sohn: Braunschweig, Germany, 1854.

10. Wall, L.; Christiansen, T.; Schwartz, R. L. *Programming Perl*, 2 ed.; O'Reilly: Sebastopol, CA, 1996.

11. HoTMetaL PRO, 3.0; SoftQuad, 56 Aberfoyle Cresent, Toronto, Canada M8X 2W4; email: mail@sq.com.

12. Perl, 5.004; http://www.perl.com/latest.html; Wall, L., OReilly and Assoicates, 101 Morris St. Sebastapol, CA 95472; email: lwall@netlabs.com.

13. Brenner, S. E.; Aoki, E. Introduction to CGI/Perl: Getting Started with Web Scripts; M & T Books: New York, 1996.

14. Schwartz, R. L.; Christiansen, T. *Learning Perl*, 2 ed.; O'Reilly: Sebastapol, CA, 1997.

15. Patchett, C.; Wright, M. *CGI/Perl Cookbook*; McGraw Hill: New York, 1997.

16. Herrmann, E. Teach Yourself CGI Progamming with Perl5 in a Week; Sams.net: Indianapolis, IN, 1996.

17. Matt's Script Archive; http://www.worldwidemart.com/scripts/; Wright, M., Worldwidemart.com, Fort Meyers, FL; email: mattw@worldwidemart.com.

18. York, D. *Can. J. Phys.* **1966**, *44*, 1079.