Designing a Web-based Interface for Student Peer Review on a Unix Server. By Joshua M. Duke[1] and Jeff Whisler[2], Department of Food and Resource Economics and IT-User Services, University of Delaware. FREC Research Report No. 02-02.

Abstract

This report describes an application of and the procedures for developing a web-based interface on a Unix server, using a simple guestbook program. The advantage of the guestbook platform is that it is commonly available on college campuses and can be secured. The application facilitates problem-based learning and other active-learning goals in an undergraduate seminar in environmental law. This report provides an example of the application and reviews the programming necessary to accomplish the learning goals.

Keywords: Problem-based Learning, Peer Review, Guestbook, Unix

Code: The computer code for this project was posted to the following web address: http://www.udel.edu/learn/jwhisler/code

Funding and Acknowledgements: This project was made possible through competitively awarded grants from the Institute for Transforming Undergraduate Education and The Present's Technological Assistance Grant, both from the University of Delaware. Suggestions and technical assistance were provided by John Mackenzie, Paul Hyde, Gabriele Bauer, and John Hall.

---

[1] Assistant Professor, Department of Food and Resource Economics, University of Delaware, Newark, Delaware 19717-1303.
[2] Technical Consultant, IT-User Services, University of Delaware, Newark, Delaware 19716.

This report describes an application of and the procedures for developing a web-based interface in a Unix environment, using a simple guestbook program. The advantage of the guestbook platform is that it may be available in a pre-packaged form on college campuses and can be secured. The application facilitates problem-based learning and other active-learning (collectively, PBL) goals in the University of Delaware undergraduate seminar, "Topics in Environmental Law", which Duke teaches. The report offers, first, a description of the course's PBL structure and goals and, second, an example of the application. The third section specifies the technical details for replicating the applications.

1. A Summary of the Course

The undergraduate seminar, "Topics in Environmental Law", includes a large PBL component and is taught yearly for fifteen to twenty students. The syllabus acts as an extensive course contract, specifying expectations about student effort, active learning exercises, and group activities. Students are divided into five permanent groups on the first day of class and each class meeting involves active-learning exercises such as 50-minute problems, role-playing advocacy, group reflection, and peer review. Content objectives center on learning the institutions and processes of environmental conflict resolution in the United States. As important, however, is the course objective to improve the analytical skills of the students. To achieve the latter objective, the course has been designed to include a semester-long problem focusing on student analysis and the incorporation of peer and instructor reviews.

Students select an actual environmental conflict to study at the beginning of the semester by searching Lexis-Nexis, using topics of interest to them. The problem, then, is to identify the fundamental conflict of interests among stakeholders and to assess the relative performance of the conflict resolution processes. Although ultimate student achievement has generally been outstanding in the past, a great deal of resistance (or lack of confidence) from students remained because of the burdens of what was seen as a 25-page research paper. To attenuate this resistance, the active-learning components of this problem were emphasized by the construction of a web-based interface for peer review.

The interface was designed so students could submit six assignments using a web form over the course of a semester. These assignments, collectively, constitute the semester-long paper. Using the interface, however, the assignments are available to all students for peer review. Group members are required to review four of the six assignments of their peers and submit at least three substantive comments for each using the web interface. In total, each student posts six assignments on the web and receives four web-based critiques from each group member. The instructor also graded and commented on each assignment. At the end of the semester, a revision of the problem is due, which requires students to assimilate between 14 and 18 separate documents of peer and instructor feedback. The technological challenges in designing and maintaining a secure, private interface were substantial.

The effect of the PBL activities on student learning has been positive. The students' groups functioned well, and the students treated one another with respect. The class had few disruptions, and the students were consistently engaged. The web-based interface encouraged the students to complete much more work out of the classroom. In addition, students tended to take more ownership of their projects, improved their level of argumentation, and seemed more accountable for the quality of their work. These quality improvements are likely due to the effect of peer pressure; when the students' problem analyses and peer reviews are posted for all the class to see, the incentive to produce a higher quality product is quite strong. The students actively collaborated in discussing and solving difficulties they were having in completing their problems. The major shortcoming in the PBL approach is that the group structure created a vehicle for organized dissent during the particularly difficult times during the semester. Overall, however, the instructor believes the students learned more, spent more time learning, enjoyed class more, and produced higher-quality projects. The higher quality work was especially evident among the lower-performing students.

2. The Application

The following figures offer examples of actual web pages (without personally identifiable student information) used in the interface. These pages were accessible only to enrolled students and the professor via a password (established separately using the University of Delaware's password options). Figure 1 shows the "gateway" to the interface, which is called the "Problems Page". Here students are divided into their semester-long groups, with whom they also share a table with during class time. By clicking on any student name, they can read or comment on that student's problem. They may also link to submit and to view their own work from this page.

Figure 1: Problems Page

Student names on the “Problems Page” link to any student’s individual problem. An example of the “Student Problem Page” is shown in figure 2. In the yellow boxes, any student or the professor can read submitted work. Each student’s problem has six sections, which are submitted over the course of the semester using the links under the “Author: Submit Information” column heading. When students click those links, they are taken to a modified guestbook form, such as the example in figure 3. Figure 3 also shows the embedded HTML formatting tags—inserted by the professor—to ensure that all student problems have the same ultimate appearance: fonts type, size, and headings. Once students post their assignments I, III, V, and VI, their group members are required to submit peer reviews. By clicking on the link in the far right column of the reviewee’s problem page, a peer reviewer is taken to a form such as figure 4. The peer review form allows as many comments to be submitted as the reviewers wish to provide. The output is simply appended to previous reviews. These reviews are accessible from the reviewee’s problem page, using the link directly below each of the four reviewed sections.

Figure 2: Student Problem Page

Figure 3: Sample Form for Submitting Problems

Figure 4: Sample Form for Submitting Peer Review

3. Technical Details

Only two separate perl programs designed by the authors must be run to launch the PBL interface.

1. startup.pl - Auto-create program

2. gb.pl – formats the guestbook file into proper HTML

A class-PBL directory is defined, and then, to initiate the program, run the startup.pl file. The program automatically builds the necessary files. It copies the files, sets the permissions, and sets the "at" job to begin one hour later. During the "at" job, the gb.pl script is run automatically. Then the "at" job is repeated every hour.

3a. What Files are Created and What They Do

The total files involved in the interface are:

1. startup.pl – main PERL file to create files and folders
   - Reads information one line at a time for each student. Reads in folder name (i.e. username), first name, last name, project title, and group number from an input file
   - Creates folders named for folder names read in
   - Sets permissions on all the files necessary for the Guestbook CGI files to work correctly.
   - Changes "main.htm" in the students' folders from the template to include their name, the title of their conflict, and a link to their supplementary files
   - Outputs a log file (default name: "log.file")
   - Outputs a the problem-index HTML page (default name: "450problems.htm")
   - Outputs a the file to be ran by the "at" command every hour (default name: "runfiles.dat")

2. gb.pl – PERL file to read from the Guestbook files and update all submitted work to the proper HTML page
   - Uses the runfiles.dat file that was created by running startup.pl and updates all information from the Guestbook files in each student's folder.

3. runfiles.dat – text file, created by startup.pl, that contains all the files to be updated by the gb.pl update script (every hour by the "at" command)

4. log.info – verbose installation information, will also say what errors are present, if any occurred

5. 450problems.htm – main HTML file to point to; all student folders are linked to from this page

6. Also required: A guestbook program. One guestbook program is supplied by the University of Delaware IT Department and is located on the central server. If anyone from another university tries to implement the program, they will need to have a similar guestbook program.

7. An input file (see section 3.c)

3b. Preliminaries Before You Start

You can run the startup.pl program from any folder, with a few requirements:

- It needs to have a folder called "templates" in the same directory containing all the files to be copied.
- It copies the files into a directory called "pbl" (default name). So, this must also be in the same directory as startup.pl. If it is not, it will be created … just be aware.
- Due to restrictions on some of the code, this program must be run and setup in the /www/htdocs/ UNIX path hierarchy. (For example, in "/www/htdocs/FREC/duke")
- The gb.pl file must reside in the same directory containing startup.pl

3c. Step-by-Step Instructions for Use

1. Upload startup.pl, gb.pl, and templates folder to the desired Copland directory

2. Make sure that startup.pl and gb.pl have executable access. If either does not, type "chmod 700 startup.pl" and/or "chmod 700 gb.pl" to give it execute access.

3. Create an input file containing the folder name and student name, one student per line in the following format:
   Filename::Student First Name::Student Last Name::Conflict Title::Group Number
   For example the file could look like:
   jwhisler::Jeff::Whisler::Conflict Title 1::1
   duke::Josh::Duke::Conflict Title 2::2

4. The input file must be ordered by increasing group number. (i.e. all group 1 students first, followed by the group 2 students, etc.)

5. To setup the folders and files: type "startup.pl". When prompted for the name of the input file, enter it and hit Enter.

6. When the program stops and says "Done," all of the folders have been created, all of the files have been placed within them, and all of the permissions have been

set. All the pages in the "pbl" directory should be able to be seen on the web. In addition, each student is allowed to ftp files into their *Student Submitted Supplementary Material* ("supp") folder. They must logon to an FTP client and then they can copy files into their "supp" folder. The "supp" folder is located at: "/www/htdocs/…etc.../pbl/USERNAME/supp" where USERNAME is equal to the student's username and the "…etc…" is equal to the path that the pbl directory is created in.
(i.e., "/www/htdocs/FREC/duke/pbl/student_username/supp")

Two FTP client options available from the University of Delaware's software distribution page are:

- WS_FTP, for Windows users, available at http://www.udel.edu/sw/win/ws_ftp_le.htm, or

- Fetch, for Mac users, available at http://www.tsc.udel.edu/macsoftdist/fetch.html

After a student FTP's a file to their directory, and the "at" script is run (within an hour), that file will be viewable by everyone. In order to stop the "at" files from being executed:

- Type: "at –l" to show the at files running and their associated job_id.

- Delete any running at files by typing: "at –r job_id"

4. Program Code

This section contains the program codes for the two files: gb.pl and startup.pl.

4.a gb.pl - Formatting Perl Program

```
#!/opt/bin/perl
while (<>){
  if(/^##$/){
        print("<hr>\n");
  }
  if (/^(.*)=(.*)$/){
         $label = $1;
         ($data = $2) =~ s/\\n/<br\>\n/g;
         print("<b>$label</b><br>\n$data\n<br>");
  }
    }
```

4.b startup.pl - Autocreate Perl Program

```
#!/opt/bin/perl
################################################################################
# Creating TextArea Forms on Copland webserver
# File: startup.pl
# Faculty Member: Josh Duke
# Programmer: Jeff Whisler: jwhisler@udel.edu
# Copyright (c) 2001, University of Delaware
#
# This script copies and sets permissions on student directories for FREC 450.
#
# Requirements:
#      1) The template directory ($templatedir) must be in the same
#         directory as this file.
#      2) The gb.pl ($perlfile) file must reside in the same directory
#         with this file
#      3) Installation needs to occur somewhere in the /www/htdocs
#         hierarchy on copland, strauss, or mahler
#
# To use:
#      1) Type: "startup.pl"
#      2) When prompted, enter the name of the input file
#
#      3) To stop the automatically running "at" program, do the following:
#         a) type "at -l" to get the process number
#         b) type "at -r process_number", (replacing "process_number" with
#            the process number you found in step a).
#
# Modifications and Use:
#      Feel free to use this code, free of charge, just give credit where
#      credit is due. The University of Delaware encourages information
#      sharing. If you improve this code, please send your improvements to:
#      jwhisler@udel.edu
#
#      If you get this code to work at another University, please let us know
#      what you have found it useful for. We like to know when our code finds
#      a use at other Universities.
#
################################################################################


use File::Copy;
use Cwd;

# files found in the template directory
@files = ('1db.db','1f.dat','1i.html','1o.html','2db.db','2f.dat',
        '2i.html','2o.html','3db.db','3f.dat','3i.html','3o.html',
        '4db.db','4f.dat','4i.html','4o.html','5db.db','5f.dat',
        '5i.html','5o.html','6db.db','6f.dat','6i.html','6o.html',
        'gb.pl','main.htm','pr1db.db','pr1f.dat','pr1i.html',
        'pr1o.html','pr3db.db','pr3f.dat','pr3i.html','pr3o.html',
        'pr5db.db','pr5f.dat','pr5i.html','pr5o.html','pr6db.db',
```

```
        'pr6f.dat','pr6i.html','pr6o.html');

@files2 = ('1i.html','2i.html','3i.html','4i.html','5i.html','6i.html',
        'pr1i.html','pr3i.html','pr5i.html','pr6i.html');

@files3 = ('1db.db','2db.db','3db.db','4db.db','5db.db','6db.db',
        'pr1db.db','pr3db.db','pr5db.db','pr6db.db');

@files4 = ('1o.html','2o.html','3o.html','4o.html','5o.html','6o.html',
        'pr1o.html','pr3o.html','pr5o.html','pr6o.html');

# Variables that may be changed:
  # Will create all user folders within '$directory' folder
    $directory = "pbl";
  # Will create install log file named '$logfile'
    $logfile = "log.info";
  # Template Directory to copy files from
    $templatedir = "templates";
  # Perl file to read guestbook file and print to static html page
    $perlfile = "gb.pl";
  # Template "main.htm" file
    $templatefile = "main.htm";
  # Template "450 rproblems.htm" file
    $templateprobfile = "templateprob.htm";
  # Temporary output file 1
    $tempfile1 = "tempmain.htm";
  # Temporary output file 2
    $tempfile2 = "tempprob.htm";
  # Temporary output file 3
    $tempfile3 = "temp.dat";
  # Temporary output file 4
    $tempfile4 = "tempprob4.htm";
  # Final main.htm file
    $finalfile1 = "main.htm";
  # Final problem main html file
    $finalfile2 = "450problems.htm";
  # Final problem page title
    $grouppagetitle = "FREC 450 Problems";
  # Final "At" command file
    $finalfile3 = "runfiles.dat";

  # Next groupnumber
    $nextGroupnumber = 0;
  # Current Working Directory
    $wholepath = getcwd;
    $curr_dir = substr($wholepath,12);

  # Variable to test if entered the sub functions yet
    $enteredSetGroups = 0;
    $enteredCreateAt = 0;

  # Variable to test if have entered into the
    $group1first = 0;
    $group2first = 0;
    $group3first = 0;
    $group4first = 0;
    $group5first = 0;
    $group6first = 0;

  # Variable to show if there was a group 6 (example/test group)
    $isGroup6 = 0;


print "\nEnter the name of the input file: ";
$filename = <STDIN>;

# Open logfile output filehandle
unless (open(LOGFILE,"> $logfile")) {
        print STDERR "Can't open file $logfile: $!\n";
}
# Print timestamp to log file
$now_string = localtime;
print LOGFILE "$now_string\n";


# Open input filehandle
unless (open(INFILE,$filename)) {
        print STDERR "Can't open file $filename: $!\n";
        print LOGFILE "Can't open file $filename: $!\n";
        exit 1;
}
```

```
# Test to see if $directory (base directory) exists; if not, create
unless (opendir(DIRHANDLE,$directory)) {
        print STDERR "$directory/ directory does Not exist ... creating ... ";
        print LOGFILE "$directory/ directory does Not exist ... creating ... ";
        # create $directory if does NOT exist
        unless (mkdir("$directory",0705)) {
                print STDERR "Error creating file $directory/: $!\n";
                print LOGFILE "Error creating file $directory/: $!\n";
                exit 2;
        }
        unless (chmod(0755,"$directory")) {
                print STDERR "Error setting permissions on directoy $directory: $!\n";
                print LOGFILE "Error setting permissions on directoy $directory: $!\n";
        }
        print "created\n";
        print LOGFILE "created\n";
}

# print encouraging message (will take a little time to copy/setup files):
print "Setting up Folders and Files ... Please wait ...\n";


# Loop through input file and create directories
while($_ = <INFILE>) {
        # Read in variables from input file
        ($username,$firstname,$lastname,$conflicttitle,$groupnumber)=split(/::/, $_);

        # Setting up Group html page
        #setGroups($lastname,$groupnumber);
        setGroups();

        # Creating new directory for student: '$directory/$username'
        unless (mkdir("$directory/$username",0705)) {
                print STDERR "Error creating file $directory/$username: $!\n";
                print LOGFILE "Error creating file $directory/$username: $! --
Exiting.\n";
                print STDERR "\nPlease make sure you want to overwrite
$directory/$username.\nIf so, please delete the folder first. (\"rm -r
$directory/$username\")\n";
                print LOGFILE "\nPlease make sure you want to overwrite
$directory/$username.\nIf so, please delete the folder first. (\"rm -r
$directory/$username\")\n";
                print "Exiting\n\n";
                print LOGFILE "Exited";
                print "View \"$logfile\" for further install information.\n\n";
                exit 3;
        }
        print LOGFILE "Creating folder: $directory/$username\n";

        # Setting CORRECT permissions
        unless (chmod(0705,"$directory/$username")) {
                print STDERR "Error setting permissions on directoy $username: $!\n";
                print LOGFILE "Error setting permissions on directoy $username: $!\n";
        }
        print LOGFILE "Setting permissions on $directory/$username\n";

        # Copying files into student directory
        foreach $i (@files) {
                unless (copy("$templatedir/$i","$directory/$username/")) {
                        print STDERR "Error copying file $i: $!\n";
                        print LOGFILE "Error copying file $i: $!\n";
                }
                print LOGFILE "copying file into $directory/$username: $i\n";
        }

        # Open main html file (main.htm) for the students
        unless (open(MAINFILE,"$templatedir/$templatefile")) {
                print STDERR "Can't open file $templatedir/$templatefile: $!\n";
                print LOGFILE "Can't open file $templatedir/$templatefile: $!\n";
                exit 1;
        }

        # Create Temporary file to temporarily store main.htm
        unless (open(TEMPFILE1,"> $directory/$username/$tempfile1")) {
                print STDERR "Can't open file $directory/$username/$tempfile1: $!\n";
                print LOGFILE "Can't open file $directory/$username/$tempfile1: $!\n";
        }

        foreach $i (<MAINFILE>) {
```

```
                # Substitute "CONFLICTTITLE" with the name of the student's conflict
                $i =~ s/CONFLICTTITLE/$conflicttitle/;

                # Substitute "STUDENTNAME" with the student's name from the infile
                $i =~ s/STUDENTNAME/$firstname $lastname/;

                print TEMPFILE1 $i;
        } # END foreach

        # Finally mv the temp file to the final filename
        `mv -f $directory/$username/$tempfile1 $directory/$username/$finalfile1`;

        close TEMPFILE1;
        close MAINFILE;


# Change the guestbook.db files links #
        # Changing links in "*i.html" files
        $k=0;
        foreach $j (@files2) {


          # Open TEMP file (each file to be changed)
          unless (open(TEMPFILEHANDLE,"$directory/$username/$j")) {
                  print STDERR "Can't open file $directory/$username/$j: $!\n";
                  print LOGFILE "Can't open file $directory/$username/$j: $!\n";
                  exit 1;
          }

          # Create Temporary file to temporarily store main.htm
          unless (open(TEMPFILE2,"> $directory/$username/tempfile2")) {
                print STDERR "Can't open file $directory/$username/tempfile2: $!\n";
                print LOGFILE "Can't open file $directory/$username/tempfile2: $!\n";
          }

          foreach $k (<TEMPFILEHANDLE>) {
                  # Substitute "CHANGE_DIR" with the current working directory
                  $k =~ s/CHANGE_DIR/$curr_dir\/$directory\/$username/;

                  print TEMPFILE2 $k;
          }

          # mv the temp file to the final
          `mv -f $directory/$username/tempfile2 $directory/$username/$j`;
          #`chmod 644 $directory/$username/$j`;

          close TEMPFILE2;
          close TEMPFILEHANDLE;

          createAt();
          $k++;
        } # END foreach


        # Setting permissions on student files
        `chmod 606 $directory/$username/*.db`;
        `chmod 644 $directory/$username/*.dat`;
        `chmod 700 $directory/$username/$perlfile`;
        `chmod 644 $directory/$username/$templatefile`;
        `chmod 644 $directory/$username/*i.html`;
        `chmod 755 $directory/$username/*o.html`;

        # Create "supp" directory for students to put work
        `mkdir $directory/$username/supp`;
        `chmod 705 $directory/$username/supp`;
        `setfacl -m user:$username:rw-,mask:rw- $directory/$username/supp`;

        # Setting FACL for folders so students can write to their "supp" folder
        `setfacl -m user:$username:--x,mask:--x $directory/$username`;
        `setfacl -m user:$username:--x,mask:--x $directory`;
        `setfacl -m user:$username:--x,mask:--x .`;

} # END WHILE


# Add closing code to and then Close TEMPFILEs
  if ( $isGroup6 == 1) { # yes, there is a group 6
    print TEMPFILE4 "\n        </table>\n      </div>\n\n  </tr>\n</table>\n\n";
    print TEMPFILE4 "<p>\n\n</body>\n</html>\n";
    close TEMPFILE4;
    print TEMPFILE3 "at -f $finalfile3 now + 1 hour\n";
```

```
    close TEMPFILE3;
  }

  elsif ( $isGroup6 == 0 ) { # no, there is not a group 6
    print TEMPFILE4 "\n        </table>\n      </div>\n    <td>\n       \n
</td>\n  </tr>\n</table>\n\n";
    print TEMPFILE4 "<p>\n\n</body>\n</html>\n";
    close TEMPFILE4;
    print TEMPFILE3 "at -f $finalfile3 now + 1 hour\n";
    close TEMPFILE3;
  }

# Finally mv the temp file to the final filename ("450problems.htm")
  `mv -f $tempfile4 $finalfile2`;
  `chmod 604 $finalfile2`;
  `mv -f $tempfile3 $finalfile3`;
  `chmod 600 $finalfile3`;

print "Done\n\n";
print LOGFILE "Done";

print "View \"$logfile\" for further install information.\n\n";

close LOGFILE;
close INFILE;

`at -f $finalfile3 now + 1 minute`;

#########################
## END Main Program
#########################


#########################
## START sub Functions
#########################


sub setGroups {

    # Test if have not modified temp file already
    if($enteredSetGroups == 0) {
        # Create Temporary file Tempfile2 (tempprob.htm) to temporarily store
"450problems.htm"
        unless (open(TEMPFILE4,"> $tempfile4")) {
            print STDERR "Can't open file $tempfile4: $!\n";
            print LOGFILE "Can't open file $tempfile4: $!\n";
        }

        $nextGroupnumber = $groupnumber + 1;

        print TEMPFILE4 "
<html>
<head>
  <title>
    $grouppagetitle
  </title>
</head>
<body>
\n<center>
  <h2>Welcome to the Problems Page.</h2>
</center>
\n<p>\n
<center>
  Here you will be able to access your problem and other problems in the class.
</center>
\n<p>\n
<center>
  To complete a peer review, read your group member's entry and then comment using their
problem page.
</center>
\n<p>\n
<table width=\"70%\" border=2 align=\"center\" bgcolor=\"#FFFF66\"
bordercolor=\"#0000FF\" cellpadding=5>
  <tr>
    <td valign=\"top\">\n";

        close TEMPFILE4;

        $enteredSetGroups = 1;
    } # END if
```

```
    if ($enteredSetGroups == 1) {
        # Open template html page ("templateprob.htm") for the problems and append to it
        unless (open(TEMPFILE4,">> $tempfile4")) {
            print STDERR "Can't open file $tempfile4: $!\n";
            print LOGFILE "Can't open file $tempfile4: $!\n";
            exit 1;
        }
    } # END if



    if ( &isSame($groupnumber,$nextGroupnumber) ) {
      if ($groupnumber!=4) {
        print TEMPFILE4 "\n        </table>\n      </div>\n    </td>\n    <td
valign=\"top\">\n";
      }
      if ($groupnumber==4) {
        print TEMPFILE4 "\n        </table>\n      </div>\n";
      }

      $nextGroupnumber++;
    }


# START Group 1
    if ( $groupnumber == 1 ) {
      if ( $group1first == 0 ) {
        print TEMPFILE4 "      <div align=\"center\">\n        <table>
          <tr>
            <td>
              <div align=\"center\">
                <big><b>Group: 1</b></big>
              </div>
            </td>
          </tr>";
        $group1first = 1;
      }
        print TEMPFILE4 "
          <tr>
            <td>
              <div align=\"center\">
                <a href=\"$directory/$username/main.htm\">$lastname</a>
              </div>
            </td>
          </tr>";

    } # END if group1


# START Group 2
    if ( $groupnumber == 2 ) {
      if ( $group2first == 0 ) {
        print TEMPFILE4 "      <div align=\"center\">\n        <table>
          <tr>
            <td>
              <div align=\"center\">
                <big><b>Group: 2</b></big>
              </div>
            </td>
          </tr>";
        $group2first = 1;
      }
        print TEMPFILE4 "
          <tr>
            <td>
              <div align=\"center\">
                <a href=\"$directory/$username/main.htm\">$lastname</a>
              </div>
            </td>
          </tr>";

    } # END if group2


# START Group 3
    if ( $groupnumber == 3 ) {
      if ( $group3first == 0 ) {
        print TEMPFILE4 "      <div align=\"center\">\n        <table>
          <tr>
            <td>
```

```
              <div align=\"center\">
                <big><b>Group: 3</b></big>
              </div>
            </td>
          </tr>";
        $group3first = 1;
      }
        print TEMPFILE4 "
          <tr>
            <td>
              <div align=\"center\">
                <a href=\"$directory/$username/main.htm\">$lastname</a>
              </div>
            </td>
          </tr>";

    } # END if group3


# START Group 4
    if ( $groupnumber == 4 ) {
      if ( $group4first == 0 ) {
        print TEMPFILE4 "    </td>\n  </tr>\n  <tr>\n    <td>\n      <div
align=\"center\">\n        <table>
          <tr>
            <td>
              <div align=\"center\">
                <big><b>Group: 4</b></big>
              </div>
            </td>
          </tr>";
        $group4first = 1;
      }
        print TEMPFILE4 "
          <tr>
            <td>
              <div align=\"center\">
                <a href=\"$directory/$username/main.htm\">$lastname</a>
              </div>
            </td>
          </tr>";

    } # END if group4


# START Group 5
    if ( $groupnumber == 5 ) {
      if ( $group5first == 0 ) {
        print TEMPFILE4 "      <div align=\"center\">\n        <table>
          <tr>
            <td>
              <div align=\"center\">
                <big><b>Group: 5</b></big>
              </div>
            </td>
          </tr>";
        $group5first = 1;
      }
        print TEMPFILE4 "
          <tr>
            <td>
              <div align=\"center\">
                <a href=\"$directory/$username/main.htm\">$lastname</a>
              </div>
            </td>
          </tr>";

    } # END if group5

# START Group 6
    if ( $groupnumber == 6 ) {
      if ( $group6first == 0 ) {
        print TEMPFILE4 "      <div align=\"center\">\n        <table>
          <tr>
            <td>
              <div align=\"center\">
                <big><b>Example:</b></big>
              </div>
            </td>
          </tr>";
        $group6first = 1;
```

```
        $isGroup6 = 1;
      }
        print TEMPFILE4 "
          <tr>
            <td>
              <div align=\"center\">
                <a href=\"$directory/$username/main.htm\">$lastname</a>
              </div>
            </td>
          </tr>";

    } # END if group6

} # END setGroups()



sub createAt {

    # Test if have not modified temp file already
    if($enteredCreateAt == 0) {
        # Create Temporary file Tempfile2 (tempprob.htm) to temporarily store
"450problems.htm"
        unless (open(TEMPFILE3,"> $tempfile3")) {
            print STDERR "Can't open file $tempfile3: $!\n";
            print LOGFILE "Can't open file $tempfile3: $!\n";
        }

        print TEMPFILE3 "$wholepath/$perlfile $directory/$username/$files3[$k] >!
$directory/$username/$files4[$k]\n";

        $enteredCreateAt = 1;
    } # END if

    elsif ($enteredCreateAt == 1) {
        # Open template html page ("templateprob.htm") for the problemsand append to it
        unless (open(TEMPFILE3,">> $tempfile3")) {
            print STDERR "Can't open file $tempfile3: $!\n";
            print LOGFILE "Can't open file $tempfile3: $!\n";
            exit 1;
        }
    } # END if


    print TEMPFILE3 "$wholepath/$perlfile $directory/$username/$files3[$k] >!
$directory/$username/$files4[$k]\n";

} # END createAt()



sub isSame {
    local ( $a, $b ) = @_ ;

    if($a == $b) {
      return 1;
    }
    elsif($a != $b) {
      return 0;
    }

} # END isSame()
```