```
foreach $key (keys %delete_list) {
  my $dir = $build_photo_path;
  my (@files) = ();
  &header;
  opendir(DIR, $dir) || &err("can’t open : $!");
  @files = grep { /$key\.*/i } readdir(DIR);
  closedir(DIR);
  if ($#files > -1) {
    foreach (@files) {
      unlink("$dir/$_") || &err("can’t delete $_ : $!");
    }
  }
}
```

```
opendir(DIR, $build_photo_path) || &err("can’t open $build_photo_path: $!")
;
@all_files = readdir(DIR);
closedir(DIR);

foreach $key (keys %delete_list) {
  &header;
  foreach (grep { /^$key\./i } @all_files) {
    unlink("$build_photo_path/$_") || &err("can’t delete $_ : $!");
  }
}
```

```
#!/usr/local/bin/perl5 -w

$input1 = 'chaintest.scan';
$output= "OUT";
$output2= "OUT.out";

open (INFILE,"$input1")||die "cannot open $input1";
open(OUTFILE,">$output")||die "canoot\n";
open(OUTFILE2,">$output2")||die "canoot\n";

#######################################################
        &preprocess;
open (INFILE2,"$output")||die "cannot open $input1";
        $/=";";
        $ct_scanout = 0;
        while (<INFILE2>){
        chomp;
        $ct_scanout = 1 if (/apply\s*\"grp[0-9]_unload\"/);
        $chain_test=1  if (/CHAIN_TEST/);

if (( /\t*chain\s+\"chain([0-9])\"/) && ($chain_test)){
                $chain_number = $1;
                &cleanup;
                        $chain_input = (split /=/,$_)[1];
                        $chain_input =~ tr/\"//d;
                SWITCH: {
                                $chain_number==1 && do {
                                @chain1 = split (//,$chain_input);
                                $scan_chain_length1 = @chain1; };
                                $chain_number==2 && do {
                                @chain2 = split (//,$chain_input);
                                $scan_chain_length2 = @chain2; };
                                $chain_number==3 && do {
                                @chain3 = split (//,$chain_input);
                                $scan_chain_length3 = @chain3; };
                                $chain_number==4 && do {
                                @chain4 = split (//,$chain_input);
                                $scan_chain_length4 = @chain4; };
                                $chain_number==5 && do {
                                @chain5 = split (//,$chain_input);
                                $scan_chain_length5 = @chain5; };
                                $chain_number==6 && do {
                                @chain6 = split (//,$chain_input);
                                $scan_chain_length6 = @chain6; };
                                $chain_number==7 && do {
                                @chain7 = split (//,$chain_input);
                                $scan_chain_length7 = @chain7;
                                &printout;
                                };
                }#END SWITCH
        }
} #end of While statement
sub printout {
        if ($ct_scanout ){
                        for ($i=0;$i<$scan_chain_length3;$i++){
               $chain1[$i] =~ s/0/L/g; $chain1[$i] =~ s/1/H/g;
               $chain2[$i] =~ s/0/L/g; $chain2[$i] =~ s/1/H/g;
               $chain3[$i] =~ s/0/L/g; $chain3[$i] =~ s/1/H/g;
               $chain4[$i] =~ s/0/L/g; $chain4[$i] =~ s/1/H/g;
               $chain5[$i] =~ s/0/L/g; $chain5[$i] =~ s/1/H/g;
               $chain6[$i] =~ s/0/L/g; $chain6[$i] =~ s/1/H/g;
               $chain7[$i] =~ s/0/L/g; $chain7[$i] =~ s/1/H/g;
                print OUTFILE2 "\n(ct_so
$chain1[$i]$chain2[$i]$chain3[$i]$chain4[$i]$chain5[$i]$chain6[$i]$
chain7[$i]
 )";
```

```
                                        }
                $ct_scanout=0;
        }
        elsif ($ct_scanout==0){
                        for ($i=0;$i<$scan_chain_length3;$i++){
                $chain1[$i] =~ s/X/0/g; $chain2[$i] =~ s/X/0/g;
                $chain3[$i] =~ s/X/0/g; $chain4[$i] =~ s/X/0/g;
                $chain5[$i] =~ s/X/0/g; $chain6[$i] =~ s/X/0/g;
                $chain7[$i] =~ s/X/0/g;
                        print OUTFILE2 "\n(ct_si  $ct_si{tdi}
$chain1[$i]$chain2[$i]$chain3[$i]$chain4[$i]$chain5[$i]$chain6[$i]$chain7[$i]
 )";
                                        }
        }
}
sub cleanup{
                  s/^\s+|\s*\n+$//g;
                  tr/\t  //d;
                  s/\n//g;
                  s/\\//g;
}
sub preprocess{
        while (<INFILE>){
                chomp;
        s/$/;/g if (/apply/);
        print OUTFILE ("$_\n");
        last if ( /^SCAN_CELLS/);
        }
        close OUTFILE;
}
```

```
#!/usr/local/bin/perl5 -w

$input1 = 'chaintest.scan';
$output= "OUT";
$output2= "OUT.out";

open (INFILE,"$input1")||die "cannot open $input1";
open(OUTFILE,">$output")||die "canoot\n";
open(OUTFILE2,">$output2")||die "canoot\n";

#######################################################
&preprocess;
open (INFILE2,"$output")||die "cannot open $input1";
$/=";";
$ct_scanout = 0;
while (<INFILE2>) {
  chomp;
  $ct_scanout = 1 if (/apply\s*\"grp[0-9]_unload\"/);
  $chain_test=1  if (/CHAIN_TEST/);

  if ((/\t*chain\s+\"chain([0-9])\"/) && ($chain_test)){
    $chain_number = $1;
    &cleanup;
    $chain_input = (split /=/,$_)[1];
    $chain_input =~ tr/\"//d;
    $chain[$chain_number] = [split //, $chain_input];
    &printout if $chain_number == 7;
  }
}

sub printout {
  if ($ct_scanout ){
    for ($i=0;$i<@{$chain[3]};$i++){
      for (@chain) {
        $_->[$i] =~ s/0/L/g;
        $_->[$i] =~ s/1/H/g;
      }
      my @chars = map $_->[$i], @chain;
      print OUTFILE2 "\n(ct_so\n", join("", @chars), "\n )";
    }
    $ct_scanout=0;
  } elsif ($ct_scanout==0) {
    for ($i=0;$i<@{$chain[3]};$i++) {
      for (@chain) {
        $_->[$i] =~ s/X/0/g;
      }
      my @chars = map $_->[$i], @chain;
      print OUTFILE2 "\n(ct_si  $ct_si{tdi}\n", join("", @chars), "
\n )";
    }
  }
}

sub cleanup {
  s/^\s+|\s*\n+$//g;
  tr/\t  //d;
  s/\n//g;
  s/\\//g;
}

sub preprocess {
  while (<INFILE>){
    chomp;
    s/$/;/g if (/apply/);
    print OUTFILE ("$_\n");
    last if ( /^SCAN_CELLS/);
  }
  close OUTFILE;
}
```

```perl
#!/usr/local/bin/perl5 -w

$input = 'chaintest.scan';
$output= "OUT.out";

open (INFILE,"$input")||die "cannot open $input";
open(OUTFILE,">$output")||die "canoot\n";

#######################################################
$/=";";
$ct_scanout = 0;

while (<INFILE>) {
  chomp;
  last if /^SCAN_CELLS/;
  $seen_CHAIN_TEST=1  if /CHAIN_TEST/;
  next unless /apply/ && $seen_CHAIN_TEST;
  die "Missing ';' in 'apply' line" if /apply[^;]*\n/;

  $ct_scanout = 1 if /apply\s*"grp[0-9]_unload"/;
  next unless /\t*chain\s+"chain([0-9])"/;
  $chain[$1] = cleanup($_);
  &printout if $1 == 7;
}

sub printout {
  if ($ct_scanout ) {
    for ($i=0;$i<@{$chain[3]};$i++){
      for (@chain) {
        next unless defined;
        $_->[$i] =~ s/0/L/g;
        $_->[$i] =~ s/1/H/g;
      }
      my @chars = map $_->[$i], @chain;
      print OUTFILE "\n(ct_so\n", join("", @chars), "\n )";
    }
    $ct_scanout=0;
  } elsif ($ct_scanout==0) {
    for ($i=0;$i<@{$chain[3]};$i++) {
      for (@chain) {
        next unless defined;
        $_->[$i] =~ s/X/0/g;
      }
      my @chars = map $_->[$i], @chain;
      print OUTFILE "\n(ct_si  $ct_si{tdi}\n", join("", @chars), "\
n )";
    }
  }
}

sub cleanup {
  local $_ = shift;
  s/.*=//s;
  tr/01X//cd;
  return [ split // ]
}

```

```
#!/usr/local/bin/perl5 -w

$input = 'chaintest.scan';
$output= "OUT.out";

open (INFILE, $input) || die "cannot open $input: $!";
open (OUTFILE,"> $output") || die "cannot open $output: $!";

$/=";";

while (<INFILE>) {
  chomp;
  last if /^SCAN_CELLS/;
  $seen_CHAIN_TEST=1  if /CHAIN_TEST/;
  next unless /apply/ && $seen_CHAIN_TEST;
  die "Missing ';' in 'apply' line" if /apply[^;]*\n/;

  my $ct_scanout = /apply\s*"grp[0-9]_unload"/;
  next unless /\t*chain\s+"chain([0-9])"/;
  $chain[$1] = cleanup($_);
  if ($1 == 7) {
    printout($ct_scanout, @chain) ;
    @chain = ();
  }
}

sub printout {
  my ($scanout, @chain) = @_;
  for my $i (0 .. $#{$chain[1]}){
    my $chars = "";
    for (@chain) {
      next unless defined;
      $chars .= $_->[$i];
    }
    if ($scanout) {
      $chars =~ tr/01/LH/;
    } else {
      $chars =~ tr/X/0/;
    }
    print OUTFILE
      $scanout ? "\n(ct_so\n" : "\n(ct_si  $ct_si{tdi}\n",
        "$chars\n )";
  }
}

sub cleanup {
  local $_ = shift;
  s/.*=//s;
  tr/01X//cd;
  return [ split // ]
}
```

```perl
#!/usr/bin/perl
#

#
#A very basic tic-tac-toe program (the computer chooses randomly)
#


use strict;
use CGI;
use Socket;

my ($rounds, $round_temp, $squares, $page, $x, $y, $z, $cell, $player_move, @available_choices, $computer_move, @choices, $round, $winner, $player_move_pretty, $computer_move_pretty);
my ($round_minus_one);  #bug fix (was recording moves for round1 as round2

$page = CGI->new();

print $page->header;
print $page->start_html();

# print table beginnings

print ("<table width=\"90\%\" border=0 cellpadding=15>\n");
print ("<tr valign=middle>\n");

# left cell is tic tac toe table

print ("<td align=center>\n");

#
# find out which round it is so we know how to define $squares
#

unless ($page->param('round')) {
   $round = 0;
}else {
   $round = $page->param('round');
}

#
# set array of tic tac toe squares
#

if ($round > 0) {
  $squares = [
            [
             $page->param('[0][0]'),
             $page->param('[0][1]'),
             $page->param('[0][2]')
            ],
            [
             $page->param('[1][0]'),
             $page->param('[1][1]'),
             $page->param('[1][2]')
            ],
            [
             $page->param('[2][0]'),
             $page->param('[2][1]'),
             $page->param('[2][2]')
            ]
           ];
}else {
   $squares = [
               ['?', '?', '?'],
               ['?', '?', '?'],
               ['?', '?', '?']
              ];
}

#
# set array for determing history of moves (recorded by round)
#

if ($round > 0) {
   $rounds = {
              round1 => {
                         player => $page->param('round1_x'),
                         computer => $page->param('round1_o')
                        },
              round2 => {
                         player => $page->param('round2_x'),
                         computer => $page->param('round2_o')
                        },
              round3 => {
                         player => $page->param('round3_x'),
                         computer => $page->param('round3_o')
                        },
              round4 => {
                         player => $page->param('round4_x'),
                         computer => $page->param('round4_o')
                        },
              round5 => {
                         player => $page->param('round5_x'),
                         computer => $page->param('round5_o')
                        }
             };
}else {
   $rounds = {
               round1 => {
                          player => '?',
                          computer => '?'
                         },
               round2 => {
                          player => '?',
                          computer => '?'
                         },
               round3 => {
                          player => '?',
                          computer => '?'
                         },
               round4 => {
                          player => '?',
                          computer => '?'
                         },
               round5 => {
                          player => '?',
                          computer => '?'
                         }
              };
}

#
# increment $round to give it a new hidden value
#

$round = $round + 1;
$round_minus_one = $round - 1;
print ("Round is: $round<br>\n");
```

```

##
## get player move using subroutine (subroutine stores it to $squares array)
##

$player_move = $page->param('choice');
if ($player_move) {
   $round_temp = "round" . $round_minus_one;
   player_moves($player_move, $page, $squares);
   $player_move_pretty = make_move_pretty($player_move);
   $rounds->{$round_temp}->{'player'} = $player_move_pretty;

#
# evaluate for winner after player moves
#

   $winner = evaluate_board($squares);
   if ($winner eq "x") {
      print ("<font color='blue'>Player Won!</font><p>\n");
      print_final_table($squares, $page, $winner, $round, $rounds);
   }else {

#
# check to see if player won. if player won, don't do the rest of this



#
# get available choices for computer choices
#

      @available_choices = get_available_choices($squares);

#
# get computer move
#
      $computer_move = get_computer_choice(@available_choices);
      ($x, $y) = split(/:/, $computer_move);
         #get coordinates for computer move and store in $x and $y
      $squares->[$x][$y] = "o";   ## change square to "o"
      $round_temp = "round" . $round_minus_one;
      $computer_move_pretty = make_move_pretty($computer_move);
      $rounds->{$round_temp}->{'computer'} = $computer_move_pretty;

   } #matches else {

} #matches if ($player_move)


#
# now that we have the array with computer and player choices, see if there is a winner
#
$winner = evaluate_board($squares);
if ($winner eq "o") {  #we already checked for x before
   print ("<font color = 'blue'>Computer Won!</font><p>\n");
   print_final_table($squares, $page, $winner, $round, $rounds);
}

if (($winner ne "o") and ($winner ne "x")) {
   print ("<font color='blue'>No winner yet</font><p>\n");

```

```
  # start form

     print $page->startform(-method=> 'POST');

  #print hidden values in form

     print_hidden_values($page, $squares, $rounds);

  # print hidden value for $round

     print "<input type='hidden' name = 'round' value='$round'>\n"
;

  # print hidden values for saving rounds


  # start tic tac toe table

     print ("<table border=1 cellpadding=10>\n<tr>");

  #
  # look through array elements ($squares) and find x's or o's
  # for all squares with no value, print a checkbox with the coord
inates
  # in the form of [0][0] as its name
  #

     foreach $x(0..2) {
        if ($squares->[0][$x] eq "?") {
           print_cell($squares, $page, $x, "0");
        }else {
           print ("<td>" . $squares->[0][$x] . "</td>\n");
        }
     }

     print ("</tr><tr>\n");
     foreach $x(0..2) {
        if ($squares->[1][$x] eq "?") {
           print_cell($squares, $page, $x, "1");
        }else {
           print ("<td>" . $squares->[1][$x] . "</td>\n");
        }
     }

     print ("</tr><tr>\n");
     foreach $x(0..2) {
        if ($squares->[2][$x] eq "?") {
           print_cell($squares, $page, $x, "2");
        }else {
           print ("<td>" . $squares->[2][$x] . "</td>\n");
        }
     }

     print ("</tr></table>");

  #
  # print warning about picking multiple squares
  #

     print $page->submit();

     print ("<p><font color='red'>Note: if you pick more than one
square, your choice will be the upper and leftmost square that you choose!!<
/font><p>\n");


```

```
      print $page->endform();
   }

   # end table cell

   print ("</td>\n");
   print ("<td align=middle>\n");

   # get printable versions of moves and print choices

   if (($player_move) or ($computer_move)) {
      foreach $x(1..$round_minus_one) {
         $round_temp = "round" . $x;
         print ("<p><b>Round $x:</b><br>\n");
         print ("\tplayer: " . $rounds->{$round_temp}->{'player'} .
 "<br>\n");
         print ("\tcomputer: " . $rounds->{$round_temp}->{'computer
'} . "<br>\n");
      }
   }else {
      print ("No moves yet.\n");
   }

   # end table

   print ("</td></tr></table>\n");

   print $page->end_html();


   sub player_moves {
      my $move = $_[0];  ##get move
      my $page = $_[1];  ## import page object
      my $squares = $_[2];  ## import array so we can change square
      my ($x, $y, $z);

      foreach $x(0..2) {
         foreach $y(0..2) {         ##test for each square
            $z = "[$x][$y]";
            if ($move eq $z) {
               $squares->[$x][$y] = "x";   ## define array element
for choice
            }
         }
      }
   }

   sub print_cell {
      my $squares = $_[0];  ##import semi-existant array of squares
      my $page = $_[1];     ##import html stuffs
      my $x = $_[2];        ##import the number that the foreach is
 on
      my $row = $_[3];      ##import the row we are on
      my $cell = "[$row][$x]";  ##get square coordinates for use in
 naming checkbx

      print ("<td>");
      print ("<input type='checkbox' name='choice' value='$cell'>\n
");
      print ("</td>");
   }

   sub get_available_choices {
      my $squares = $_[0];
      my ($x, $y, $z);
      my @available_choices = ();
```

```

       foreach $x(0..2) {
          foreach $y(0..2) {
             unless (($squares->[$x][$y] eq "x") or ($squares->[$x][$y] eq "o")) {
                 $z = "$x:$y";    ## return in this form for later use (so we can split it by the colon)
                 push (@available_choices, $z);
             }
          }
       }
       return @available_choices;
    }

    sub get_computer_choice {
       my @available_choices = @_;
       my $length = @available_choices;
       my $number = int(rand() * ($length - 1));
       my $choice = $available_choices[$number];
       return $choice;  ## this will be a coordinate, in the form of $x:$y from $z above
    }


    sub print_hidden_values {
       my $page = $_[0];
       my $squares = $_[1];
       my $rounds = $_[2];
       my ($x, $y, $cell, $round);

       #print hidden values for cells
       foreach $x(0..2) {
            foreach $y(0..2) {
               $cell = "[$x][$y]";
               print ("<input type=’hidden’ name=’$cell’ value=’" . $squares->[$x][$y] . "’>");
               print ("\n");
            }
       }

       #print hidden values for rounds (history)
       foreach $x(1..5) {
          $round = "round" . $x;
          print ("<input type=’hidden’ name=’$round" . "_x’ value =’" . $rounds->{$round}->{’player’} . "’>\n");
          print ("<input type=’hidden’ name=’$round" . "_o’ value=’" . $rounds->{$round}->{’computer’} . "’>\n");
       }
    }


    sub evaluate_board {
        my ($squares) = $_[0];
        my ($x, $y, $winner);

        foreach $x (0..2) {
            if (
                ($squares->[$x][0] eq $squares->[$x][1]) and
                ($squares->[$x][1] eq $squares->[$x][2])
               ) {
                $winner = $squares->[$x][0];
                return $winner;
            }
        }
        foreach $y (0..2) {
            if (
                ($squares->[0][$y] eq $squares->[1][$y]) and
                ($squares->[1][$y] eq $squares->[2][$y])
                ) {
                $winner = $squares->[0][$y];
                return $winner;
            }
        }
        if (
            (($squares->[1][1] eq "x") or ($squares->[1][1] eq "o"))
            and
             (
              (($squares->[0][0] eq $squares->[1][1]) and
               ($squares->[1][1] eq $squares->[2][2]))
              or
              (($squares->[0][2] eq $squares->[1][1]) and
               ($squares->[1][1] eq $squares->[2][0]))
             )
            ) {
            $winner = $squares->[1][1];
            return $winner;
        }
    }

    sub print_final_table {
       my $squares = $_[0];
       my $page = $_[1];
       my $winner = $_[2];
       my $round = $_[3];
       my $rounds = $_[4];
       my ($visitor, $visitor_name, $time);

       ## print ending table

       print $page->startform(action=>'tic_tac.cgi',
                              method=>'POST'
                             );

       print_hidden_values($page,$squares,$rounds);

       print ("<input type='hidden' name='round', value='$round'>\n");

       print ("<table border=1 cellpadding=10>\n");
       print ("<tr valign=middle>\n");
       foreach $x(0..2) {             ##print first row
          print ("<td align=center>" . $squares->[0][$x] . "</td>\n");
       }
       print ("</tr><tr valign=middle>\n");
       foreach $x(0..2) {             ##print second row
          print ("<td align=center>" . $squares->[1][$x] . "</td>\n");
       }
       print ("</tr><tr valign=middle>\n");
       foreach $x(0..2) {             ##print third row
          print ("<td align=center>" . $squares->[2][$x] . "</td>\n");
       }
       print ("</tr></table><p>\n");

       print ("<a href=\"http://soya.serve.com/cgi-bin/tic_tac.cgi\">Play Again!</a>");

       print $page->end_html();

```

```
     #print log of play

     $visitor = $page->remote_host();
     if ($visitor =~ /\d*\.\d*\.\d*\.\d*/) {
      $visitor_name = gethostbyaddr(inet_aton($visitor), AF_INET);
     }

     $time = localtime(time());

     open (MAIL, "| /usr/sbin/sendmail -t");
     print MAIL "To: author\@example.com\n";
     print MAIL "Subject: tic tac toe results\n";
     print MAIL "\n$visitor, $visitor_name: $time: $winner on round $round";
     close MAIL;
  }

  sub make_move_pretty {
     my ($move) = $_[0];  #get move (either $player_move or $computer_move
     my (%squares_names, $pretty_move);

     if ($move =~ /:/) {
        $move =~ s/^(\d):(\d)/$1$2/;
     }
     else {
        $move =~ s/^\[(\d)\]\[(\d)\]/$1$2/;
     }

     %squares_names = ("00" => "top left",
                       "01" => "top center",
                       "02" => "top right",
                       "10" => "center left",
                       "11" => "center",
                       "12" => "center right",
                       "20" => "lower left",
                       "21" => "lower center",
                       "22" => "lower right"
                      );

    $pretty_move = $squares_names{$move};
    return $pretty_move;
  }
```

```
#!/usr/bin/perl
#

#
#A very basic tic-tac-toe program (the computer chooses randomly)
#


use strict;
use CGI;
use Socket;

my ($rounds, $round_temp, $squares, $page, $x, $y, $z, $cell,
    $player_move, @available_choices, $computer_move, @choices, $round,
    $winner, $player_move_pretty, $computer_move_pretty);

my ($round_minus_one);  #bug fix (was recording moves for round1 as round2

$page = CGI->new();

print $page->header;
print $page->start_html();

# print table beginnings

print ("<table width=\"90\%\" border=0 cellpadding=15>\n");
print ("<tr valign=middle>\n");

# left cell is tic tac toe table

print ("<td align=center>\n");

#
# find out which round it is so we know how to define $squares
#

unless ($page->param(’round’)) {
   $round = 0;
}else {
   $round = $page->param(’round’);
}

#
# set array of tic tac toe squares
#

for my $x (0..2) {
  for my $y (0..2) {
    $squares->[$x][$y] = $page->param("$x$y");
  }
}

#
# set array for determing history of moves (recorded by round)
#

if ($round > 0) {
        for my $rn (1..5) {
          $rounds->{"round$rn"} =
            { player   => $page->param("round${rn}_x"),
              computer => $page->param("round${rn}_o"),
            }
        }
}

```

```
#
# increment $round to give it a new hidden value
#

$round = $round + 1;
$round_minus_one = $round - 1;
print ("Round is: $round<br>\n");

##
## get player move using subroutine (subroutine stores it to $squares array)
##

$player_move = $page->param('choice');
if ($player_move) {
   $round_temp = "round" . $round_minus_one;
   player_moves($player_move, $page, $squares);
   $player_move_pretty = make_move_pretty($player_move);
   $rounds->{$round_temp}->{'player'} = $player_move_pretty;

#
# evaluate for winner after player moves
#

   $winner = evaluate_board($squares);
   if ($winner eq "x") {
      print ("<font color='blue'>Player Won!</font><p>\n");
      print_table($squares, $page, $winner, $round, $rounds, "final");
   }else {

#
# check to see if player won. if player won, don't do the rest of this


#
# get available choices for computer choices
#

      @available_choices = get_available_choices($squares);

#
# get computer move
#
      $computer_move = get_computer_choice(@available_choices);
      ($x, $y) = split(//, $computer_move);
         #get coordinates for computer move and store in $x and $y
      $squares->[$x][$y] = "o";   ## change square to "o"
      $round_temp = "round" . $round_minus_one;
      $computer_move_pretty = make_move_pretty($computer_move);
      $rounds->{$round_temp}->{'computer'} = $computer_move_pretty;

   } #matches else {

} #matches if ($player_move)


#
# now that we have the array with computer and player choices, see if there is a winner
#
$winner = evaluate_board($squares);
if ($winner eq "o") {  #we already checked for x before
   print ("<font color = 'blue'>Computer Won!</font><p>\n");
   print_table($squares, $page, $winner, $round, $rounds, "final");
}

```

```
if (($winner ne "o") and ($winner ne "x")) {
   print ("<font color='blue'>No winner yet</font><p>\n");


print_table($squares, $page, $winner, $round, $rounds);


}

# end table cell

print ("</td>\n");
print ("<td align=middle>\n");

# get printable versions of moves and print choices

if (($player_move) or ($computer_move)) {
   foreach $x(1..$round_minus_one) {
      $round_temp = "round" . $x;
      print ("<p><b>Round $x:</b><br>\n");
      print ("\tplayer: " . $rounds->{$round_temp}->{'player'} . "<br>\n");
      print ("\tcomputer: " . $rounds->{$round_temp}->{'computer'} . "<br>\n");
   }
}else {
   print ("No moves yet.\n");
}

# end table

print ("</td></tr></table>\n");

print $page->end_html();


sub player_moves {
   my $move = $_[0];  ##get move
   my $page = $_[1];  ## import page object
   my $squares = $_[2];  ## import array so we can change square
   my ($x, $y, $z);

   foreach $x(0..2) {
      foreach $y(0..2) {         ##test for each square
         if ($move eq "$x$y") {
            $squares->[$x][$y] = "x";   ## define array element for choice
         }
      }
   }
}

sub get_available_choices {
   my $squares = $_[0];
   my ($x, $y, $z);
   my @available_choices = ();

   foreach $x(0..2) {
      foreach $y(0..2) {
         unless (($squares->[$x][$y] eq "x") or ($squares->[$x][$y] eq "o")) {
          $z = "$x$y";
          push (@available_choices, $z);
         }
      }
   }
```

```perl
      return @available_choices;
   }

   sub get_computer_choice {
      my @available_choices = @_;
      my $length = @available_choices;
      my $number = int(rand() * ($length - 1));
      my $choice = $available_choices[$number];
      return $choice;  ## this will be a coordinate, in the form of $x$y from $z above
   }


   sub print_hidden_values {
      my $page = $_[0];
      my $squares = $_[1];
      my $rounds = $_[2];
      my ($x, $y, $cell, $round);

      #print hidden values for cells
      foreach $x(0..2) {
        foreach $y(0..2) {
              $cell = "$x$y";
            print ("<input type='hidden' name='$cell' value='" . $squares->[$x][$y] . "'>");
              print ("\n");
           }
      }

      #print hidden values for rounds (history)
      foreach $x(1..5) {
         $round = "round" . $x;
         print ("<input type='hidden' name='$round" . "_x' value ='" . $rounds->{$round}->{'player'} . "'>\n");
         print ("<input type='hidden' name='$round" . "_o' value='" . $rounds->{$round}->{'computer'} . "'>\n");
      }
   }


   sub evaluate_board {
       my ($board) = $_[0];

       my @table = (
          [ 0,0 , 0,1 , 0,2 ],
          [ 1,0 , 1,1 , 1,2 ],
          [ 2,0 , 2,1 , 2,2 ],
          [ 0,0 , 1,0 , 2,0 ],
          [ 0,1 , 1,1 , 2,1 ],
          [ 0,2 , 1,2 , 2,2 ],
          [ 0,0 , 1,1 , 2,2 ],
          [ 0,2 , 1,1 , 2,0 ],
       );

       for my $win (@table) {
         my ($x1, $y1, $x2, $y2, $x3, $y3) = @$win;
         if ($board->[$x1][$y1] eq $board->[$x2][$y2]
             &&  $board->[$x1][$y1] eq $board->[$x3][$y3]) {
           return $board->[$x1][$y1];
         }
       }
       return;
   }

   sub print_table {
      my ($squares, $page, $winner, $round, $rounds, $final) = @_;
```

```
       my ($visitor, $visitor_name, $time);

       ## print ending table

       print $page->startform(-method=>'POST');

       print_hidden_values($page,$squares,$rounds);

       print ("<input type='hidden' name='round', value='$round'>\n");

       print ("<table border=1 cellpadding=10>\n");
       print ("<tr valign=middle>\n");
       for my $row (0..2) {
         for my $col (0..2) {
            my $cell = $squares->[$row][$col];
            if ($cell eq "") {
              $cell = $final ? "?"
                    : "<td><input type='checkbox' name='choice' value='$row$col'></td>\n";
            }
            print ("<td align=center>" . $cell . "</td>\n");
         }
         print ("</tr><tr valign=middle>\n") unless $row == 2;
       }
       print ("</tr></table><p>\n");

       if ($final) {
         print ("<a href=\"http://www.example.com/cgi-bin/tic_tac.cgi\">Play Again!</a>") ;
       } else {
         print $page->submit();

         print ("<p><font color='red'>Note: if you pick more than one square, your choice will be the upper and leftmost square that you choose!!</font><p>\n");
       }

       print $page->endform();


       print $page->end_html();

       if ($final) {
         #print log of play

         $visitor = $page->remote_host();
         if ($visitor =~ /\d*\.\d*\.\d*\.\d*/) {
           $visitor_name = gethostbyaddr(inet_aton($visitor), AF_INET);
         }

         $time = localtime(time());

         #   open (MAIL, "| /usr/sbin/sendmail -t");
         #   print MAIL "To: author\@example.com\n";
         #   print MAIL "Subject: tic tac toe results\n";
         #   print MAIL "\n$visitor, $visitor_name: $time: $winner on round $round";
         #   close MAIL;
       }

    }

    sub make_move_pretty {
       my %squares_names = ("00" => "top left",
                            "01" => "top center",
                            "02" => "top right",
                            "10" => "center left",
                            "11" => "center",
                            "12" => "center right",
                            "20" => "lower left",
                            "21" => "lower center",
                            "22" => "lower right"
                           );

       return $squares_names{$_[0]};
    }


```

```
#!/usr/bin/perl
#

#
#A very basic tic-tac-toe program (the computer chooses randomly)
#


use strict;
use CGI;
use Socket 'AF_INET';

sub do_with_board (&);

my ($rounds, $squares, $round, $player_move, $computer_move, $winner);

my $page = CGI->new();

print $page->header;
print $page->start_html();

# print table beginnings

print ("<table width=\"90\%\" border=0 cellpadding=15>\n");
print ("<tr valign=middle>\n");

# left cell is tic tac toe table

print ("<td align=center>\n");

#
# find out which round it is so we know how to define $squares
#

unless ($page->param('round')) {
   $round = 0;
}else {
   $round = $page->param('round');
}

#
# set array of tic tac toe squares
#

do_with_board { $squares->[$a][$b] = $page->param("$a$b") };

#
# set array for determing history of moves (recorded by round)
#

if ($round > 0) {
        for my $rn (1..5) {
          $rounds->{"round$rn"} =
            { player   => $page->param("round${rn}_x"),
              computer => $page->param("round${rn}_o"),
            }
        }
}

#
# increment $round to give it a new hidden value
#

my $round_minus_one = $round;
$round = $round + 1;
print ("Round is: $round<br>\n");
```

```
##
## get player move using subroutine (subroutine stores it to $squares array
)
##

$player_move = $page->param('choice');
if ($player_move) {
   my $round_temp = "round" . $round_minus_one;
   player_moves($player_move, $page, $squares);
   my $player_move_pretty = make_move_pretty($player_move);
   $rounds->{$round_temp}->{'player'} = $player_move_pretty;

#
# evaluate for winner after player moves
#

   $winner = evaluate_board($squares);
   if ($winner eq "x") {
      print ("<font color='blue'>Player Won!</font><p>\n");
      print_table($squares, $page, $winner, $round, $rounds, "final");
   }else {

#
# check to see if player won. if player won, don't do the rest of this


#
# get available choices for computer choices
#

      my @available_choices = get_available_choices($squares);

#
# get computer move
#
      $computer_move = $available_choices[int rand @available_choices];
      my ($x, $y) = split(//, $computer_move);
         #get coordinates for computer move and store in $x and $y
      $squares->[$x][$y] = "o";   ## change square to "o"
      $round_temp = "round" . $round_minus_one;
      my $computer_move_pretty = make_move_pretty($computer_move);
      $rounds->{$round_temp}->{'computer'} = $computer_move_pretty;

   } #matches else {

} #matches if ($player_move)


#
# now that we have the array with computer and player choices, see if there
 is a winner
#
$winner = evaluate_board($squares);
if ($winner eq "o") {  #we already checked for x before
   print ("<font color = 'blue'>Computer Won!</font><p>\n");
   print_table($squares, $page, $winner, $round, $rounds, "final");
}

if (($winner ne "o") and ($winner ne "x")) {
   print ("<font color='blue'>No winner yet</font><p>\n");


   print_table($squares, $page, $winner, $round, $rounds);
```

```
}

# end table cell

print ("</td>\n");
print ("<td align=middle>\n");

# get printable versions of moves and print choices

if ($player_move or $computer_move) {
   foreach my $x (1..$round_minus_one) {
      my $round_temp = "round" . $x;
      print ("<p><b>Round $x:</b><br>\n");
      print ("\tplayer: " . $rounds->{$round_temp}->{'player'} . "<br>\n");
      print ("\tcomputer: " . $rounds->{$round_temp}->{'computer'} . "<br>\
n");
   }
}else {
   print ("No moves yet.\n");
}

# end table

print ("</td></tr></table>\n");

print $page->end_html();


sub player_moves {
  my ($move, $page, $squares) = @_;

  ## define array element for choice
  do_with_board {$squares->[$a][$b] = "x"};
}

sub get_available_choices {
   my $squares = $_[0];
   my @available_choices = ();

   do_with_board {
     unless (($squares->[$a][$b] eq "x") or ($squares->[$a][$b] eq "o")) {
       push @available_choices, "$a$b";
     }
   };
   return @available_choices;
}



sub print_hidden_values {
  my ($page, $squares, $rounds) = @_;

   #print hidden values for cells
   do_with_board {
     print "<input type='hidden' name='$a$b' value='" . $squares->[$a][$b]
. "'>\n";
   };

   #print hidden values for rounds (history)
   foreach my $x (1..5) {
     my $round = "round$x";
     print ("<input type='hidden' name='$round" . "_x' value ='" . $rounds-
>{$round}->{'player'} . "'>\n");
     print ("<input type='hidden' name='$round" . "_o' value='" . $rounds->
{$round}->{'computer'} . "'>\n");
   }
```

```
}


sub evaluate_board {
    my ($board) = $_[0];

    my @table = (
       [ 0,0 , 0,1 , 0,2 ],
       [ 1,0 , 1,1 , 1,2 ],
       [ 2,0 , 2,1 , 2,2 ],
       [ 0,0 , 1,0 , 2,0 ],
       [ 0,1 , 1,1 , 2,1 ],
       [ 0,2 , 1,2 , 2,2 ],
       [ 0,0 , 1,1 , 2,2 ],
       [ 0,2 , 1,1 , 2,0 ],
    );

    for my $win (@table) {
      my ($x1, $y1, $x2, $y2, $x3, $y3) = @$win;
      if ($board->[$x1][$y1] eq $board->[$x2][$y2]
          &&  $board->[$x1][$y1] eq $board->[$x3][$y3]) {
        return $board->[$x1][$y1];
      }
    }
    return;
}

sub print_table {
   my ($squares, $page, $winner, $round, $rounds, $final) = @_;
   my ($visitor, $visitor_name, $time);

   ## print ending table

   print $page->startform(-method=>'POST');

   print_hidden_values($page,$squares,$rounds);

   print ("<input type='hidden' name='round', value='$round'>\n");

   print ("<table border=1 cellpadding=10>\n");
   print ("<tr valign=middle>\n");
   for my $row (0..2) {
     for my $col (0..2) {
       my $cell = $squares->[$row][$col];
       if ($cell eq "") {
         $cell = $final ? "?"
           : "<td><input type='checkbox' name='choice' value='$row$col'></t
d>\n";
       }
       print ("<td align=center>" . $cell . "</td>\n");
     }
     print ("</tr><tr valign=middle>\n") unless $row == 2;
   }
   print ("</tr></table><p>\n");

   if ($final) {
     print ("<a href=\"http://www.example.com/cgi-bin/tic_tac.cgi\">Play Ag
ain!</a>") ;
   } else {
     print $page->submit();

     print ("<p><font color='red'>Note: if you pick more than one square, y
our choice will be the upper and leftmost square that you choose!!</font><p>
\n");
   }
   print $page->endform();


   print $page->end_html();

   if ($final) {
     #print log of play

     $visitor = $page->remote_host();
     if ($visitor =~ /\d*\.\d*\.\d*\.\d*/) {
       $visitor_name = gethostbyaddr(inet_aton($visitor), AF_INET);
     }

     $time = localtime(time());

     #   open (MAIL, "| /usr/sbin/sendmail -t");
     #   print MAIL "To: author\@example.com\n";
     #   print MAIL "Subject: tic tac toe results\n";
     #   print MAIL "\n$visitor, $visitor_name: $time: $winner on round $ro
und";
     #   close MAIL;
   }

}

sub make_move_pretty {
   my %squares_names = ("00" => "top left",
                         "01" => "top center",
                         "02" => "top right",
                         "10" => "center left",
                         "11" => "center",
                         "12" => "center right",
                         "20" => "lower left",
                         "21" => "lower center",
                         "22" => "lower right"
                        );

   return $squares_names{$_[0]};
}

sub do_with_board (&) {
  my $code = shift;
  for $a (0..2) {
    for $b (0..2) {
      $code->();
    }
  }
}
```