A Web-based Examination in Programming Exercise Providing Different Problems to Each Student

KITA Toshihiro, MIYAUCHI Hajime and HIYAMA Takashi

Department of Electrical and Computer Engineering, Kumamoto University

t-kita@eecs.kumamoto-u.ac.jp

Abstract

Internet-based education has various kinds of possibilities, but we have some difficulties in making a fair online evaluation of how the each student understands the contents of the online lecture. In this paper a simple example of Web-based examination that was actually done in a programming exercise class is presented. The problems of the exam had been made by students themselves as the answers to a previous homework and are randomly given to the different students.

1. Introduction

Internet-based or IT-based education is supposed to greatly grow in number and in quality in several years. We expect many kinds of merits of IT-based education. It enables us to let learners more eager to study the subject, to easily share the world-wide instructive resources that have been created by other educational staff in the world, to give students distance learning at anywhere in the world, and many other things.

However, we also find difficulties in IT-based education. For example, it is not so easy to make a fair online evaluation of how well the students have understood the contents. Of course, it is a good compromise that we give a traditional style exam to the students instead of an online exam while we make a online lecture, but we will naturally wish to do without offline matters. There are several disturbances for realizing fair grading such as mere duplication of homework answers between the students, illegally pretending to be other persons to answer the exam or something like those.

Our Web-based exam presented in the following sections is a simple example as a solution or tips to this issue and it was actually done in a programming exercise class.

2. Objectives of the exercise class

Before we describe the Web-based exam, we like to explain the objectives of the C programming exercise class in which we did the online exam.

In the class we intend to make the students who have only used ready-made applications on their PC's interested in computer programming rather than to teach the details of the C language. To let the students practically learn how to get computers work according to what they wish to, we give them exercises of making C language programs of graphical animations or for solving puzzles. That is to learn how to convert their own ideas into computational algorithms, which is required for them to analyze data or to create new approaches in their future research activities. It also leads to their brief understanding of the computer mechanisms.

In our Web-based exam presented in this paper, we told the students to solve puzzles, particularly the puzzles called "illust-logic", using their own programs that they had prepared.

2.1. What is illust-logic puzzle?

Illust-logic is a puzzle game also known as Paint by Numbers and Pikros which is very popular in Japan. Illust-logic puzzles are easy to be made but difficult to be solved. We utilize this feature of the puzzle in doing our exam.

Figure 1 shows an example of illust-logic puzzle. When you complete the answer of the puzzle by marking the squares of the grid, some pattern or

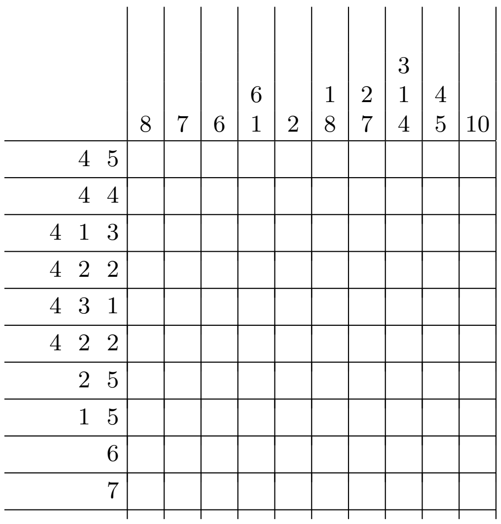


Figure 1: Example of illust-logic puzzle

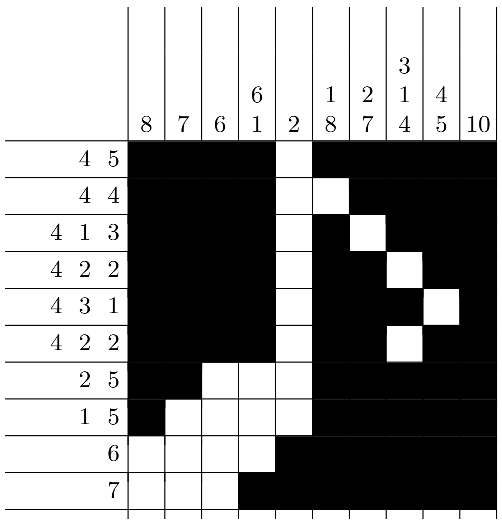


Figure 2: The answer of the puzzle

figure will appear in the grid area as shown in Figure 2.

The numbers around the grid area indicate the run-length of the points to be marked in each row or column. We call the numbers "the guiding numbers." For example, for the 10-by-10 grid as Figure 1, "10" for a column or a row means that the all the points should be marked, and "4 5" means that you should make four marked, one unmarked and five marked points. But for "2 5", you have several possible answers such as

- 2 marked, 1 unmarked, 5 marked and 2 unmarked points
- 2 marked, 2 unmarked, 5 marked and 1 unmarked points
- 2 marked, 3 unmarked and 5 marked points
- 1 unmarked, 2 marked, 1 unmarked, 5 marked and 1 unmarked points

and so on. To determine which is the right answer, you should first mark or unmark some of the points in the row or the column by considering the guiding numbers in the different (i.e., orthogonal) direction. Checking all the rows and columns to and fro repeatedly, you will get the final answer, an eighth note as in Figure 2 for this example.

You can find the explanations of how to play illust-logic puzzle in several Web pages[1] although almost all the pages are written in Japanese.

Obviously, to make the problems of illust-logic puzzles is quite easy. You have only to make some pattern and just to count the number of points joining in each row and column. But to solve the problems is difficult and time-consuming.

3. Web-based exam

In this section the Web-based exam which was actually done in a programming exercise class of our department is presented.

3.1. Outline of the exam

Our Web-based exam is based on a simple idea that students make their own problems. In advance of the exam, students are required to submit 3 types of material as their homework by e-mail; 10-by-10 and 25-by-25 patterns, a C language program which was used to generate the guiding numbers from the patterns and the generated guiding numbers. That is not only for collecting various exam problems, but also for getting the students familiar with the basic way to make C language programs since the program which generates the guiding numbers from a pattern is relatively simple.

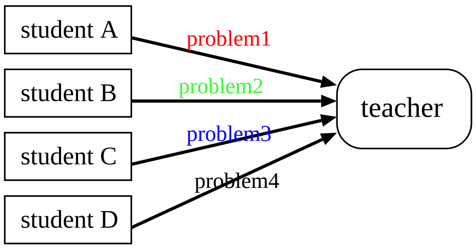


Figure 3: Students submit problems.

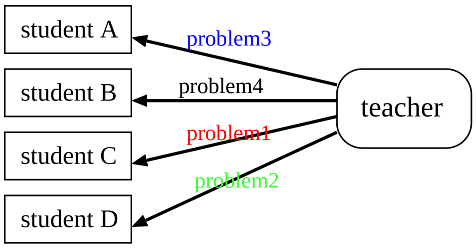


Figure 4: Examination problems are shuffled.

The submitted problems are easy to make as mentioned above and they are also difficult to solve, so they are appropriate for the exam of this style.

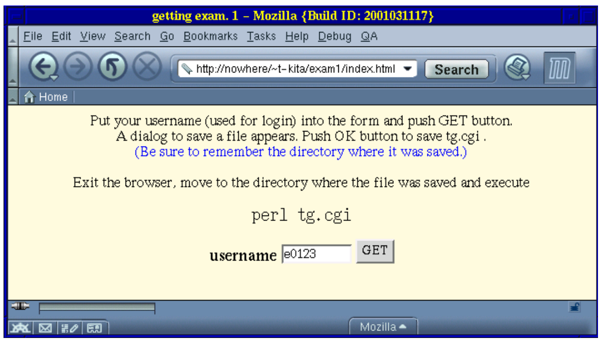
At the examination time, the collected problems are shuffled and randomly given to the students who are different from those who made the problems. The collected answers to the exam are also collected through the network i.e., each student uploads the answers to a server machine. The answers are checked and graded manually.

3.2. Procedure for each student in the online exam

The students take the following procedure in the online exam.

1. They make an access to the top Web page of the exam, enter their own usernames and push GET button to get a file. (Figure 5, List 1)

2. The browser requests to save the file (named `tg.cgi`) to the disk because the mime type is specified as application/octet-stream in the HTTP header of the CGI script. (Figure 6) The CGI script itself (located in the server) is as shown in List 2. When it is saved to the disk of users' machines it becomes again a Perl script like List 3.


getting exam. 1 – Mozilla {Build ID: 2001031117}
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://nowhere/~t-kita/exam1/index.html
Search
Home
Mozilla

Put your username (used for login) into the form and push GET button.
A dialog to save a file appears. Push OK button to save tg.cgi .
(Be sure to remember the directory where it was saved.)

Exit the browser, move to the directory where the file was saved and execute

perl tg.cgi

username e0123 GET

Figure 5: Get exam

```
<HTML><HEAD><TITLE>getting exam. 1</TITLE></HEAD>
<BODY BGCOLOR=cornsilk><font size=4><center>

Put your username (used for login) into the form
and push GET button.<br>
A dialog to save a file appears.
Push OK button to save tg.cgi .<br>
<font color=blue>
(Be sure to remember the directory where it was saved.)
</font><br><br>
Exit the browser, move to the directory where the file was saved and execute
<pre><font size=5>perl tg.cgi</font></pre>

<FORM ACTION="./tg.cgi" METHOD="POST">
<font size=5>username </font>
<INPUT TYPE="text" NAME="username" SIZE="10" VALUE="e0123">
<INPUT TYPE="submit" VALUE="GET">
</FORM>

</center></font></BODY></HTML>
```

List 1: index.html

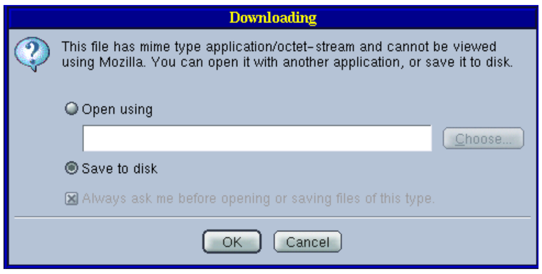
3. The downloaded script includes all the exam files. The students must execute `perl tg.cgi` to retrieve the exam problems. There appears the instruction (Figure 7) on how to make answers and how to submit them. The students try to solve the problems and to create answer files.

4. To upload the answers, they are required to execute the command `mka` (List 4) that is also one of the retrieved files. In the execution of `mka`, an archive file that includes all the files involving the answers are made and the students get access to the uploading page. The students need to push Browse button to select the archive file and to click Upload button. (Figure 8, List 5)

5. When the upload is completed by the CGI script shown in List 6, they can see the list of the files they have uploaded at the confirmation message like Figure 9.

```
#!/usr/bin/perl

$examdir= "exam1129";    # directory where the exam will be made
$datadir= "./guidedata";     # server directory where the problems are put
$URI= "http://nowhere/~t-kita/exam1";     # where cgi scripts are put
$max12= 63;   # max. number of the 10-by-10 problem files (for ex1 and ex2)
$max3=  37;   # max. number of the 25-by-25 problem files (for ex3)

use CGI_Lite;

$cgi = new CGI_Lite();
%formdata = $cgi->parse_form_data();

$username= $formdata{’username’};
$debug= $formdata{’debug’};

unless ($username =~ /e[0-9][0-9][0-9]/){
    print "Content-type: text/html", "\n\n";
    print "<font color=red size=5>Invalid username!</font>\n";
    exit(0);
}

$unum= substr($username,1); # 3-digit number in username

print "Content-type: Application/Octet-Stream", "\n\n";
printf(’$dir= "%s";’."\n",$examdir);
if ($unum ne "0000"){
    printf(’if ($ENV{"USER"} ne "%s"){ ’,$username);
    printf(’die "Invalid username!".$ENV{"USER"}." != %s"; }’."\n",$username);
}
printf(’mkdir $dir, 0700 or die "cannot create directory.";’."\n");

# embed three problems (guiding numbers of illust-logic puzzles).
# problem files are put in $datadir.
# for ex1 and ex2, the files are named like 0059-e0123g10.txt.
# for ex3          the files are named like 0034-e0123g25.txt.
# they are distinguishable by the first 4-digit number.
# (it must start from 0001 and no discontinuity is allowed.)
srand $unum; # always give the same problems to the same student
$lastselection= -1;
for($i=1; $i<=3; $i++){
    $fname= "ex".$i.".guide";  # filename to be used
    if ($i==1 or $i==2){
while(1){ # avoid to give the same problem doublly
    $selection= &sl($unum,$i);
    if ($lastselection != $selection){
$lastselection= $selection;
last;
    }
}
$globfile= sprintf("%s/%04d-*g10.txt",$datadir,$selection);
    }else{
$globfile= sprintf("%s/%04d-*g25.txt",$datadir,&sl($unum,$i));
    }
    unless (@tmp= glob($globfile)){
printf(’printf("\nError! Notify the teacher by raising your hand.\n");’);
printf(’printf("data file not found!(%s)\n");’,$globfile);
die;
    }
    $insertfile= $tmp[0];
    if ($debug eq "DEBUG"){ printf("#guidefilename= %s\n",$insertfile); }
    printf(’open FH, ">$dir/%s" or die "cannot create files.";’."\n",$fname);
    print "print FH <<END;\n";
    open GFH, "<$insertfile";
    while(<GFH>){ print $_; }
    print "END\n";
    print "close(FH);\n";
    printf (’chmod 0400, "$dir/%s";’."\n",$fname);
}

############################ mka ##########################################
# ----------------------------------------------------------------------
print << ’GEND’;
open FH, ">$dir/mka" or die "cannot create file.";
print FH <<’END’;
#! /usr/bin/perl
GEND
# ------------------------------------------------------------------------

printf(’$arc= "%s-ans.tgz";’."\n",$username);
if ($unum ne "0000"){
    printf(’if ($ENV{"USER"} ne "%s"){ ’,$username);
    printf(’die "Invalid username!".$ENV{"USER"}." != %s"; }’."\n",$username);
}

# ----------------------------------------------------------------------
print << ’GEND’;
@f= ("ex1.guide","ex2.guide","ex3.guide",  "ex1.ans","ex2.ans","ex3.ans" );
foreach (@f) {
    unless (-r $_){ die "$_ not found.";}
}
@prog= glob("*.c") or die "C-language program file not found.";
@prog= glob("*.c *.h");
$allfiles= join(" ",@f)." ".join(" ",@prog);
(system("gtar cvzf $arc $allfiles")==0) or die "tar error.";
print "\n$allfiles have been tar+gzipped as the file named \n$arc.\n";
open FH, "pwd|";
$ufile= <FH>; chomp $ufile; $ufile.="/".$arc; close(FH);
print "Invoking netscape to jump to the uploading page...\n";
GEND
# ------------------------------------------------------------------------
 printf(’system("mozilla %s/uploadform.cgi?username=%s")’."\n",$URI,$username);
# ----------------------------------------------------------------------
print << ’GEND’;
END
close(FH);
chmod 0500, "$dir/mka";
GEND
# ------------------------------------------------------------------------


############################ README #######################################
# ----------------------------------------------------------------------
print << ’GEND’;
open FH, ">$dir/README" or die "unable to create file";
print FH <<END;
******************* READ THIS CAREFULLY **********************

Move into the directory named $dir and solve the problems.

Copy all the programs (C-language files) into $dir, and
do the compilations or execute make command in this directory
to make the executable that solves the problems.

There are 3 data files of the problems to be solved :
ex1.guide
ex2.guide
ex3.guide

Try to reproduce the patterns from these guiding numbers.
The results should be put in the files named as
ex1.ans
ex2.ans
ex3.ans
with or without editors.

It is OK to include all the output.
Put the incomplete pattern if it timed-out.

To submit your answer, execute
mka
and you can upload the answer.

You can read the same instruction in README file.
END
close(FH);
open FH, "<$dir/README";
system("clear");
while(<FH>){ print $_; }
unlink $0;
GEND
# ------------------------------------------------------------------------
exit(0);


# randomly select problems for each.
# always return the same problems because "srand $unum" is executed above.
sub sl{
    local($unum, $i)= @_;
    if     ($i==1 or $i==2){
return int(rand()*($max12-0.0001))+1;
    }else{
return int(rand()*($max3-0.0001))+1;
    }
}
```

List 2: tg.cgi


Downloading

This file has mime type application/octet-stream and cannot be viewed using Mozilla. You can open it with another application, or save it to disk.

○ Open using

[✓ Save to disk]

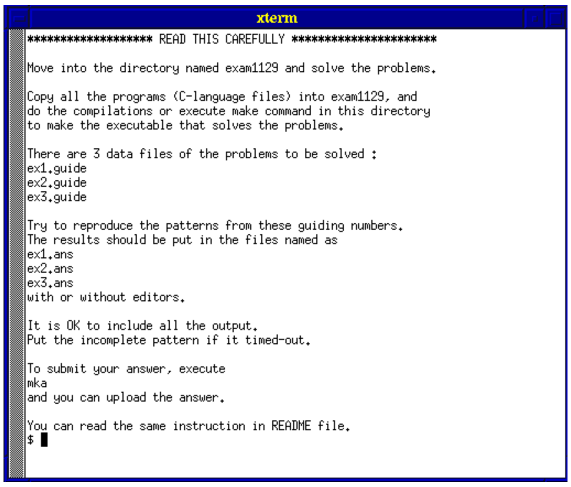
☒ Always ask me before opening or saving files of this type.

Figure 6: Save exam to disk

```
$dir= "exam1129";
if ($ENV{"USER"} ne "e0123"){ die "Invalid username!".$ENV{"USER"}." != e0123"; }
mkdir $dir, 0700 or die "cannot create directory.";
open FH, ">$dir/ex1.guide" or die "cannot create files.";
print FH <<END;
horizontal 10
2
4 2
4 2
4 2
2 7
2 6
7
6
4
4
vertical 10
3
4
2
9
10
10
9
2
5
4
END
close(FH);
chmod 0400, "$dir/ex1.guide";
open FH, ">$dir/ex2.guide" or die "cannot create files.";
print FH <<END;
horizontal 10
2 2 2
1 2 2 1
```

(cutting out the rest)

List 3: tg.cgi (downloaded)


xterm

```
****************** READ THIS CAREFULLY *********************

Move into the directory named exam1129 and solve the problems.

Copy all the programs (C-language files) into exam1129, and
do the compilations or execute make command in this directory
to make the executable that solves the problems.

There are 3 data files of the problems to be solved :
ex1.guide
ex2.guide
ex3.guide

Try to reproduce the patterns from these guiding numbers.
The results should be put in the files named as
ex1.ans
ex2.ans
ex3.ans
with or without editors.

It is OK to include all the output.
Put the incomplete pattern if it timed-out.

To submit your answer, execute
mka
and you can upload the answer.

You can read the same instruction in README file.
$ 
```

Figure 7: Retrieve exam files

```
#! /usr/bin/perl
$arc= "e0123-ans.tgz";
if ($ENV{"USER"} ne "e0123"){ die "Invalid username!".$ENV{"USER"}." != e0123"; }
@f= ("ex1.guide","ex2.guide","ex3.guide",  "ex1.ans","ex2.ans","ex3.ans" );
foreach (@f) {
    unless (-r $_){ die "$_ not found.";}
}
@prog= glob("*.c") or die "C-language program file not found.";
@prog= glob("*.c *.h");
$allfiles= join(" ",@f)." ".join(" ",@prog);
(system("gtar cvzf $arc $allfiles")==0) or die "tar error.";
print "\n$allfiles have been tar+gzipped as the file named \n$arc.\n";
open FH, "pwd|";
$ufile= <FH>; chomp $ufile; $ufile.="/".$arc; close(FH);
print "Invoking Web browser to jump to the uploading page...\n";
system("mozilla http://nowhere/~t-kita/exam1/uploadform.cgi?username=e0123")
```

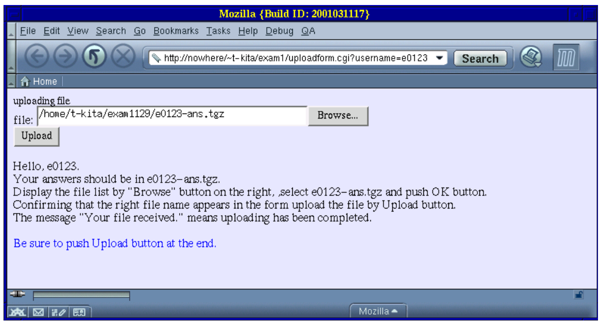
List 4: mka

The uploading function of this system has been realized thanks to the Perl module `CGI_Lite` by Shishir Gundavaram that can be found in CPAN sites[2].

4. Conclusion

An example of simple Web-based examination has been presented. As the students are given different problems from each other they were supposed to take the exam under a fair circumstance, which is expected to lead to their higher motivation to learn.

Beside the style of the examination, it is of course quite important to decide how to present and what timing to give hints and advice on the improvement


Mozilla {Build ID: 2001031117}

File Edit View Search Go Bookmarks Tasks Help Debug QA
http://nowhere/~t-kita/exam1/uploadform.cgi?username=e0123
Search
Home

uploading file

file: /home/t-kita/exam1129/e0123-ans.tgz

Hello, e0123.
Your answers should be in e0123-ans.tgz.
Display the file list by "Browse" button on the right, ,select e0123-ans.tgz and push OK button.
Confirming that the right file name appears in the form upload the file by Upload button.
The message "Your file received." means uploading has been completed.

Be sure to push Upload button at the end.

Figure 8: Upload answer files

```
#!/usr/bin/perl

use CGI_Lite;

$cgi = new CGI_Lite();
%formdata = $cgi->parse_form_data();
$username = $formdata{'username'};

print "Content-type: text/html", "\n\n";

# ----------------------------------------------------------
print << 'HT_END';
<HTML><HEAD><TITILE>uploading file</TITLE></HEAD>
<font size=4>
<BODY bgcolor=lavender>
<FORM ACTION="./upload.cgi" ENCTYPE="multipart/form-data" METHOD="POST">
file: <INPUT TYPE="file" NAME="File" SIZE="50"><BR>
<INPUT TYPE="submit" VALUE="Upload">
</FORM>
HT_END
# ----------------------------------------------------------
print << "HT_END";
Hello, $username.<br>
Your answers should be in $username-ans.tgz.<br>
Display the file list by "Browse" button on the right,
,select $username-ans.tgz and push OK button.<br>
Confirming that the right file name appears in the form
upload the file by Upload button.<br>
The message "Your file received." means uploading has been completed.
<br><br>
<font color=blue>
Be sure to push Upload button at the end.
</font>

</font>
</BODY></HTML>
HT_END

exit(0);
```

List 5: uploadform.cgi

of their own programs and algorithms.

The examination actually done for this time is in synchronous style; All the students took the exam in a computer room at the same time. The asynchronous style examination is desirable, but it will be more difficult to be done neatly.

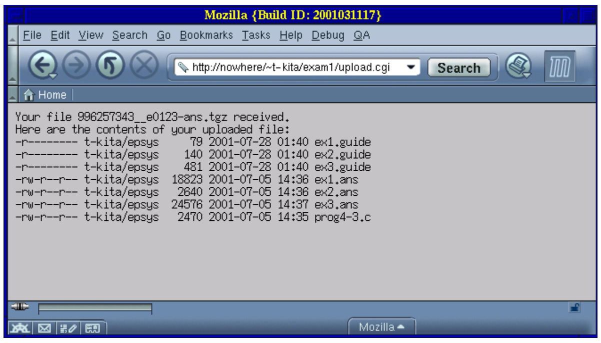
References

[1] http://www.microlink.co.jp/products/minilogic/lecture.htm

[2] http://www.cpan.org/

```
#!/usr/bin/perl

# most parts are borrowed from a CGI_Lite example file named "upload".

use CGI_Lite;

$cgi = new CGI_Lite;
$indir= "./incoming";
$cgi->set_directory ($indir) || die "Directory doesn't exist.\n";
$cgi->set_platform ("Unix");
$cgi->set_buffer_size (1024);
$cgi->filter_filename (\&my_way);
$cgi->set_file_type ('handle');
$cgi->add_mime_type ('application/mac-binhex40');
$cgi->add_mime_type ('application/binhex-40');
$cgi->remove_mime_type ('text/html');

$data = $cgi->parse_form_data;

print "Content-type: text/plain", "\n\n";

if ($data->{'File'} =~ /_$/){
    print "filename not specified";
    exit(0);
}
if ($cgi->is_error) {
    $error_message = $cgi->get_error_message;
    print <<End_of_Error;
Error! Raise your hand to notify.
$error_message
End_of_Error
}else{
    $File = $data->{File};
    print <<End_of_Header;
Your file $File received.
Here are the contents of your uploaded file:
End_of_Header
    open PR, "tar tvzf $indir/$File |";
    while (<PR>) {
        print;
    }
    $cgi->close_all_files;
}
exit (0);

sub my_way
{
    my $file = shift;

    $file =~ tr/A-Z/a-z/;                # Upper to lowercase
    $file =~ s/(?:%20)+/_/g;             # One or more spaces to "_"
    $file =~ s/%[\da-fA-F]{2}//g;        # Remove all %xx
    return ($file);
}
```

List 6: upload.cgi


Mozilla {Build ID: 2001031117}
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://nowhere/~t-kita/exam1/upload.cgi
Search
Home
Mozilla

```
Your file 996257343__e0123-ans.tgz received.
Here are the contents of your uploaded file:
-r-------- t-kita/epsys     79 2001-07-28 01:40 ex1.guide
-r-------- t-kita/epsys    140 2001-07-28 01:40 ex2.guide
-r-------- t-kita/epsys    481 2001-07-28 01:40 ex3.guide
-rw-r--r-- t-kita/epsys  18823 2001-07-05 14:36 ex1.ans
-rw-r--r-- t-kita/epsys   2640 2001-07-05 14:36 ex2.ans
-rw-r--r-- t-kita/epsys  24576 2001-07-05 14:37 ex3.ans
-rw-r--r-- t-kita/epsys   2470 2001-07-05 14:35 prog4-3.c
```

Figure 9: Confirmation of upload