Student_____

The Online Grade Book Project

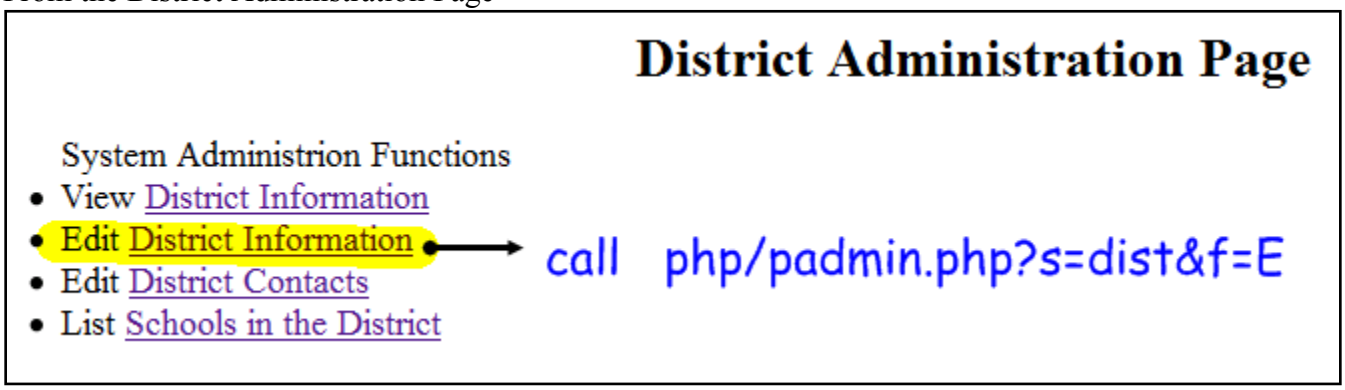
Date: _____

The to_string function – Example

The –E command line argument, along with the district number, will create a ~ separated value string that contains the major fields of the district record. Below is the command line used to execute the DistrictMgr and the output:

Command Line: ../exe/DistrictMgr -K 2 –E
Output:
2~International School of Boston~45 Matignon Road~Cambridge~MA~02140~617.499.1451~617.499.1454

How the to_string function is used:
From the District Administration Page



When the "Edit District Information" link is followed, the padmin.php program is called with two parameters:

s=dist tells the padmin.php program to use the District file as the source
f=E tells the padmin.php program to create and Edit screen.

From padmin.php:

```
switch ($s) {
  case "dist": // district file
       switch($f) { // function
         case "E": // edit
            echo "Edit district record\n";
            $output = shell_exec("../exe/DistrictMgr -K $did -E");
            list($v_id,    $v_name,  $v_address,
                 $v_city,  $v_state, $v_zip,
                 $v_phone,   $v_fax) = explode("~",$output);
            fwrite($TRACEOUT, "     DistrictMgr -E return $output\n");
            $output="";
            echo "<hr>";
            include '../html/distEdit.php';
            break;
```

The DistrictMgr program is called, and the value of the variable $output contains:
2~International School of Boston~45 Matignon
Road~Cambridge~MA~02140~617.499.1451~617.499.1454

The Online Grade Book Project

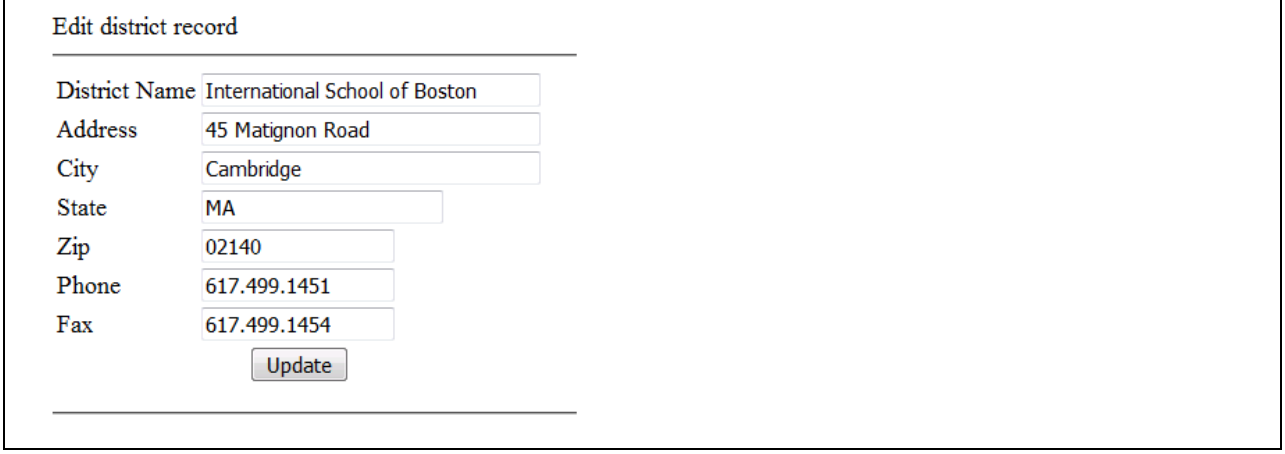
Date: _____

The php function **explode** is called to parse $output using the ~delimiters, and each of the variables in the list is set to the appropriate value.

```
list($v_id,  $v_name,  $v_address,
     $v_city,  $v_state, $v_zip,
     $v_phone,   $v_fax) = explode("~",$output);
```

When the **../html/distEdit.php** file is included, the variable name is substituted with its value:

```
<!--District Edit Form -->
<form method=post action=../php/padmin.php?s=dist&f=S>
    <input type=hidden name=did value="<?php echo "$v_id"  ?>">
<table border=0>
<tr><td>District Name </td>
    <td><input name=dname value="<?php echo "$v_name" ?>" size=30>
    </td></tr>
<tr><td>Address </td>
    <td><input name=daddr value="<?php echo "$v_address" ?>" size=30>
    </td></tr>
<tr><td>City </td>
    <td><input name=dcity value="<?php echo "$v_city" ?>" size=30>
    </td></tr>
<tr><td>State </td>
    <td><input name=dstate value="<?php echo "$v_state" ?>" size=20>
    </td></tr>
<tr><td>Zip </td>
    <td><input name=dzip value="<?php echo "$v_zip" ?>"size=15>
    </td></tr>
<tr><td>Phone </td>
    <td><input name=dphone value="<?php echo "$v_phone"?>" size=15>
    </td></tr>
<tr><td>Fax   </td>
    <td><input name=dfax value="<?php echo "$v_fax" ?>" size=15>
    </td></tr>
<tr><td colspan=2><center><input type=submit name=submit value="Update">
    </center>
    </td></tr>
</table>
</form>
```



When the variables are undefined, and file **../html/distEdit.php** is included, the form will be blank which would be suitable for adding a new District record.

When the user presses the [Update] button, padmin is called with s=dist with a function that means "Save"

```
<form method=post action=../php/padmin.php?s=dist&f=S>

case "S": // Save/Update
             $did      = $_POST["did"];"";
             $v_name   = $_POST["dname"];"";
             $v_address= $_POST["daddr"];"";
             $v_city   = $_POST["dcity"];"";
             $v_state  = $_POST["dstate"];"";
             $v_zip    = $_POST["dzip"];"";
             $v_phone  = $_POST["dphone"];"";
             $v_fax    = $_POST["dfax"];"";
             $out1="$v_name~$v_address~".
                   "$v_city~$v_state~$v_zip~$v_phone~$v_fax";
             //echo "Out1 = $out1\n";
             //echo "List of all the user-form arguments<ul>";
             //foreach ($_POST as $key => $value){
             //       echo "<li>{$key} = {$value}\r\n";
             //       }
             //echo "</ul>";
            fwrite($TRACEOUT, " Call DistrictMgr -K $did -U \"$out1 \"\n");
            $output = shell_exec("../exe/DistrictMgr -K $did -U \"$out1\"");
            fwrite($TRACEOUT, " DistrictMgr return $output\n");
            break;
```

padmin.php will take the form data to create command line arguments for the DistrictMgr to process an update command (-U).

First, each form field is copied into a local variable with a v_ prefix.
The variable $out1 is initialized with the District fields separated by a ~
The shell_exec command connects the php program to the operating system to call the district manager with all of the necessary command line arguments.
The output fromt eh DistrictMgr call is copied into a variable names $output. The fwrite calls shown above write to a log file which is used to trace and debug the DistrictMgr.

Prior to the termination of the padmin.php program, the $output is sent to the user using the echo command, and the trace file is closed:

```
echo "<hr><pre>\n$output\n</pre>\n";
 fclose($TRACEOUT);
```

The Online Grade Book Project

The command line arguments for adding records to the District file or updating the records to the district file use the ~ separated values. When the DistrictMgr receives a string that contains multiple fields separated by ~ it will call n = get_tokens(args, tokens, "~");

Where args is string of fields,
tokens is an array of strings declared as: Tokens tokens;

In this example, n is an integer variable that will contain the number of tokens retuned by get_tokens.

The Online Grade Book Project

Date: _____

Your File Manager, Version 3:
Expected completing day: Tuesday May 10, finish for homework as needed

a. Update the menu to include the command –A "field~string"
b. Update the main program to process the –A "field~~string" command
    1. Parse the "field~string" by calling get_tokens
    2. Initialize a file structure from the tokens
    3. Use the FTOC to find out what the new Key should be, and use that value to update the file structure.
    4. Write out the structure to the end of the data file
    5. Update the FTOC with the next record number, and increment the number of records in the file – and write out the updated FTOC.
c. Add in appropriate program traces to verify that your program is correct:
    1. Before adding the record, print the TOC if Debug > 1
    2. After the record is initialized from the argument string, print out the contents of the record by calling a function.
    3. After the TOC is updated and written back to the file, print it out the FTOC.
d. Update the main menu to process the –K n command. When this command line argument is processed a variable should be set to the value of n which will contain the "Key" to your file. Example, if you are processing students, the Key will be a student ID, in the DistrictMgr the key is a district ID.
e. Create a function that can locate a specified record in your file, return the contents of the record, and set the file location to be at the start of the record (in case it gets updated, it can be written exactly where the file was left rather than needing to use fseek again to find the record). Here is an example from the DistrictMgr:

    int lock_filedistrict(FTOC ftoc,District *dist, int KEY, FILE *fp) {
    This function takes four parameters
        1) the FTOC to the District file, which has already been read in
        2) a pointer to a District record, where the data is read into
        3) a KEY to look for, it must match a district ID
        4) A file to search through – the file must be opened already
    If the record is found, the return code will have the value of the KEY.
    If the record is not found, the return code will be 0.
    If there is an error reading the file the return code will be -1.

f. Create a –K n –R command, that will locate a record with Key n, and print its contents. If a user issues a –R command without first specifying –K n, then print the contents of all records in the file that have a key value > 0.
g. Update the menu to include –R command (R for Report)
h. Create a to_string function that will take a single argument (a structure) and return a string that contains all of the fields in the structure separated by the "~" character.
i. Create a command –E to export the contents of a specified record by calling to_string.
j. Update the menu to include the command –K n -E"