Student_____

The Online Grade Book Project

Date: _____

Table of Contents

Requirements _____ ***2***

File genfiles.c _____ ***5***

File: GenData,c _____ ***5***

File school.c _____ ***5***

File coursemgr.c _____ ***6***

File departmgr.c _____ ***7***

File districtmgr.c _____ ***8***

File authmgr.c _____ ***9***

The Online Grade Book Project

Date: _____

Requirements

Each project team should create a separate file manager program to control each aspect of the following files:

a) Students.ibcs struct Student
b) Teachers.ibcs struct Teacher
c) Classes.ibcs struct Class
d) Assignments.ibcs struct Assignment
e) Grades.ibcs struct Grade

Each group can partition the work up as they see fit, however, **each student must be responsible for creating at least one each of the following project components**:

a) an HTML form that passes data through to a PHP program, that then access one of the data files.
b) a C function that finds a specified record in one of the data files.
c) a C function that prints out the contents of a record in one of the data files.
d) a C function that performs a logical delete of a record in one of the data files.
e) a C function that adds a record to one of the data files, and updates the FTOC
f) a C function that finds data in one (or more) of the data files and returns a string that becomes integrated into an HTML form, or report page.

Each file manager must contain a function that performs the following tasks, and uses only the values that are passed by using parameters. The ***_FILESTRUCT*** part of the function name represents one of the 5 files for Student, Teacher, Class, Assignment, and Grade

int prt_FILE(***FILESTRUCT*** var);
Requirement: Accepts a variable that is one of the file struct, and use printf to display the contents to the screen. This function will be used mostly for debugging and interactive program tracing.

char *to_string_FILE(***FILESTRUCT*** var, char flag);
Requirement: Accept a variable that is one of the file stuct and return a string that contains the contents of the record. DO NOT use printf, all of the information must be contained in the return string. if the value of flag is equal to 'H' then return the string in HTML format. You might find it helpful to create other values for flag to customize how the string is returned. You also might consider an flag value to indicate the field values should be returned in a single string separated by ~ or another standard symbol.

int init_FILE(***FILESTRUCT*** *var, char *args);
Requirement: Accept the address of a file structure, so that the structure can be updated and returned to the calling module, and update the contents of the struc with the values in the args array.
The return value should be 1 if the struct was updated successfully and 0 otherwise. An error code should be returned if the number of arguments in args is not what you expected.

Example:

```
n = get_tokens(args, tokens, "~");
if (n != 5 ) {
   ret = 0; // not ok
   }
 else { var -> field1 = atoi(tokens[0]);
        strcpy(var->field2,    tokens[2]);
      …}
```

int lock_FILE(FTOC ftoc, ***FILESTRUCT*** *var, int KEY, FILE *fp) ; {
Requirements: Loop through the specified file looking for a record that matches the specified KEY value. Use the ***FILESTRUCT * var*** parameter as your read buffer, and use the FTOC ftoc to determine the maximum number of records to look through. If the record has been located in the file, then set the file pointer to the start of the record – if the calling module updates the record after locking it, the calling program can simply "write" the record back to the current location. [If you do not use fseek to backup the file pointer you would end up overwriting data that was unrelated to the record found.] If the record is found, return the value of the KEY. If the record is not found, return 0; if there was a file error using fread, fwrite, of fseek then return -1.

File Update Notes

a) When you are accepting command line arguments for an Add, you will not accept the primary key. New records will be assigned in order using the information that is stored in the FTOC. You need to use the nextRec field to get the key for the next record, and update the nextRec field when you add new records to the data file.
b) When you delete a record from the file we will only perform a "logical delete." To perform a logical delete you will multiply the ID for the deleted record by -1, this will identify both that the record has been delete and will contain the content of the record as it was prior to being deleted.
c) When you send a report of all of the records in a file, be sure to send only data from valid and active records, and no information from deleted records.
d) When you are looping through the file using the FTOC record count, remember that will be the total number of records in the file, deleted as well as active records.

The Online Grade Book Project

Date: _____

You may write additional functions as needed to keep the code in the main program manageable. Below in the pseudocode that the main file manager program is required to support – you may added additional features as you need. You can look at the existing file managers (DeptMgr, SchoolMgr) for guidance.

int main(int argc, char *argv[]) {

Loop through and select the command line arguments
when "-help" display a menu of options similar to:

FILESTRUCT Manager Command Line Options:

-help:	Display this list
-TOC:	Displays File TOC
-R:	Displays all records
-K <key> -R	Displays information about specified record
-A <string>:	Adds a record
-D	Deletes a record, -K also required
-K <key>:	Specified a file KEY
-U <string>	Updates a record, -K also required
-ListNames	Display a list of records -K group-id required

when "-R" process a report/print command
if no KEY has already been specified
then loop through all of the records in the file
call prt_ ***FILESTRUCT***
if a KEY had been specified
then call lock_ ***FILESTRUCT*** and then prt_ ***FILESTRUCT***

when "-ListNames" create a list of the records in the file that can be used in an HTML select statement.

when "-TOC" display the FTOC for this file
print_ftoc(my_ftoc); // this can be found in ibcsprocs.obj

when "-D" perform a logical delete – multiply the ID key field by -1 to mark it as deleted – all deleted records will remain in the data file. Update the deleted record in the file.

when "-K" the user is providing a record key which will be used in other commands

when "-A" process an **Add** request, all of the fields that are needed to add a record are supplied on the next command line argument using a string that is delimited by "~".

When "-U process an **Update** request, all of the fields that are needed to update a record are supplied on the next command line argument using a string that is delimited by "~". The –K id parameter should have already been specified.
lock the specified record, replace all of the fields that are specified in the next argument.

The Online Grade Book Project

//File genfiles.c

```
//Executable file        exe/GenFiles
//Name: Mr.Brennan
//Purpose: Arguments -I : generate a set of empty files
//                   -S : show the sizeof each file structure
//                -help : show this list
```

//File: GenData,c

```
//Executable file        exe/GenData
//Purpose : populate some of the Online Grade Book Tables
//  Start from district 10
//  a) Add some school file records          ./GenData      -Schools
//  b) Add some department file records      ./GenData      -Departments
//  c) add some course file records          ./GenData      -Courses
```

//File school.c

```
//Executable file        exe/SchoolMgr
//Name: Mr.Brennan
//Purpose: Data functions centered around the School table
        School Manager Command Line Options:
        -help:          Display this list
        -TOC:           Displays File TOC
        -R:             Displays all records
        -K <key> -R     Displays information about specified school
        -A <string>:    Adds a record
        -D              Deletes a record, -K also required
        -K <key>:       Specified a School ID (Key)
        -U  <string>    Updates a record, -K also required
        -ListNames      Display a list of schools, -K district also required
```

To Compile the School Manager

```
gcc schoolmgr.c ../lib/ parser_traces_off ../lib/ibcsprocs_no_traces.obj -I ../h -o
../exe/SchoolMgr
```

The Online Grade Book Project

//File coursemgr.c

//Executable file exe/CourseMgr
//Name: Mr.Brennan
//Purpose: Data functions centered around the Courses table

Course Manager Command Line Options:
-help:\t\t Display this list
-TOC:\t\t Displays File TOC
-R:\t\t Displays all records
-K <key> -R\t Displays information about specified course
-A <string>:\t Adds a record
-D \t\t Deletes a record, -K also required
-K <key>:\t Specified a Course ID (Key)
-U <string>\t Updates a record, -K also required
-ListNames \t Display a list of courses, -K department id also required
-DEBUG\t Turns on some program tracing

To Compile the Course Manager

```
gcc  coursemgr.c   ../lib/parser_traces_off
../lib/ibcsprocs_no_traces.obj -I ../h -o ../exe/CourseMgr
```

To Compile the DeparMgr Manager

```
gcc  departmgr.c   ../lib/parser_traces_off
../lib/ibcsprocs_no_traces.obj -I ../h -o ../exe/DepartMgr
```

The Online Grade Book Project

//File departmgr.c

//Executable file exe/DepartmentMgr
//Name: Mr.Brennan
//Purpose: Data functions centered around the Department table

Department Manager Command Line Options:

-help:	Display this list
-TOC:	Displays File TOC
-R:	Displays all records
-K <key> -R	Displays information about specified department
-A <string>:	Adds a record
-D	Deletes a record, -K also required
-K <key>:	Specified a Department ID (Key)
-U <string>	Updates a record, -K also required
-ListNames	Display a list of departments -K school id

To Compile the DeparMgr Manager

```
gcc  departmgr.c   ../lib/parser_traces_off
../lib/ibcsprocs_no_traces.obj -I ../h -o ../exe/DepartMgr
```

The Online Grade Book Project

Date: _____

//File districtmgr.c

//Executable file exe/DistrictMgr
//Name: Mr.Brennan
//Purpose: Manage the District file of the Online Grade Book Project

District Manager Command Line Options:

-help:	Display this list
-TOC:	Displays File TOC
-R:	Displays all records
-R: -K <key>	Displays information about specified district
-A <string>:	Adds a record
-D	Deletes a record, -K also required
-K <key>:	Specified a District ID (Key)
-U <string>	Updates a record, -K also required
-C # <string>	Updates contact list, -K also required
-ListNames	Generate a select list of District names for the login form

To Compile the District Manager

```
gcc  districtmgr.c ../lib/parser_traces_off
../lib/ibcsprocs_no_traces.obj -I ../h -o ../exe/DistrictMgr
```

To Add District Data

Syntax ./DistrictMgr -A "name~address~city~state~phone~fax~zip".

Sample Data – Basic Records

../exe/DistrictMgr -A "Watertown Publuc Schools~30 Common Street~Watertown~MA~617-926-7700~no fax~02472"

../exe/DistrictMgr -A "International School of Boston~45 Matignon Road~Cambridge~MA~617.499.1451~617.499.1454~02140"

Add userid/password

../exe/DistrictMgr -K key -U n "userid~password"
../exe/DistrictMgr -K 2 -U 0 "ibcsid1~ibcspw1"
../exe/DistrictMgr -K 2 -U 1 "ibcsid2~ibcspw2"

Rcommended Changes

The District record needs a **home page** field, somewhere to go when a District userid/password is validate or for a HOME link on the distruct web page.
Workaround: the validate.php will redirect based on valid district ids

The Online Grade Book Project

//File authmgr.c

//Executable file exe/AuthMgr
/-/Name: Mr.Brennan
//Purpose: Perform all userid validated for the Online Grade Book Project

To compile the authmgr

```
gcc  authmgr.c   ../lib/parser_traces_off
../lib/ibcsprocs_no_traces.obj -I ../h -o ../exe/AuthMgr
```

AuthMgr Command Line Options:

Validate a distrtic id
../exe/AuthMgr -D 2 Returs 0 not found, or the Did if it is valid

Validate a District userd from the Distrtic.ibcs file
../exe/AuthMgr –T A –U userid –P password –D Did
-T specifies the Validation Type
"A" for district admin
"S" for students
"T" for teachers
Returs 0 not found, or the Did if it is valid

Validate a user from the Teacher file, -D represents a school iD
../exe/AuthMgr –T T –U userid –P password –D schooled
Returns 0 for an invalid userid/password/school id combination
Returms the sid (student id) for a valid userid/password/school id combination

Validate a user from the Students file, -D represents a school iD
../exe/AuthMgr –T S –U userid –P password –D schooled
Returns 0 for an invalid userid/password/school id combination
Returms the sid (student id) for a valid userid/password/school id combination