CSC 300
Assignment # 2
200 Points
Due: TBA (check D2L)

The use of indexes as a way to efficiently access large amounts of data is well understood. Indexes are a very important part of any modern Relational Database Management System (RDMS). When used with random-access files, an index can be viewed as a lookup table for the location of records within the file. Each record is logically identified using its record id (RID). The RID value corresponds to a records location in the file. For a file of R records, the first record would have a RID of 0, the last would have a RID value of R-1. The ability to randomly access a record within a file through the use of its RID removes the need to sequentially search the file record by record!

Indexes and Random Access Files

Working directly with data from secondary storage in an efficient manner is made easier through the use of **Random-Access File** processing. This method allows us to access items in a file much like we do in a single-dimensional array. The file will be logically looked at as being made up of homogeneous records of the same size (in bytes). We can then use a logical record number in the same way we use a subscript value in an array. By pre-calculating the starting location (RID) of a record in the file, we can directly access 1 record's worth of binary data. An **index** will hold a value used to identify the record and the RID for the location of that record within the file. By using this index, we will be able to know the RID of the record we are looking for in advance. (***see random access handout & index handout***)

For this assignment you will first use the program developed (reviewed) in class to create a file of employee records. Each record will have a unique employee id number. Next you will design and implement a menu-driven program to allow the user to work with the employee records in an efficient manner.

Program End-User requirements.

(1) The program will build an internal index of the records in the file **empl.dat** using the **empid** field as the key for more efficient processing of the data.

(2) The program will be menu driven and allow the user to :

a) VIEW an employee record by supplying their **empid** number.
b) EDIT the ***non-key*** field **deptnum** of an employee record.
c) Sequentially view all employee records ***sorted in ascending empid order***.

(3) The program should work with files that have **up to** 100 employee records in then.

==
OUTPUT: ***{see the output section later in this document}***
==
Design Requirements

You should create a Global constant to hold the path to the BINARY file being used in the program.

Your program design should use a menu driven interface.

Your program should include (***at least***) the following functions as part of the design:

{note: these are just general descriptions of the functions so may need more or arguments etc..}

1) A function that will be passed in an empty index and will open the **empl.dat** file, build the index on the **empid** field and then close the file and pass back the index.

2) A function that will be passed in the index and will sort the index records in ***ascending order*** by their **empid** field values and pass it back. (can choose any array sorting method0

3) A function that will open the **empl.dat** file, print the records out to the screen in ascending **empid** order, and then close the file.

   *{might be useful to have a **pre-condition** that the **index** has been already sorted ☺ }*

4) A function that will be passed in an **impid** and index will return its corresponding **RID**, found in the index.

5) A function that will be passed in a RID and will **return** the corresponding record from the **empl.dat** file. The function should open the file, seek to the correct location, read the record in, and close the file.

6) A function that will be passed in an employee record and RID and will write the record out to the **empl.dat** file. It should open the file, seek to the RID, write the record, and close the file.

==

Documentation Requirements

TBA (2014)

==

Submission Guidelines

- You should drop the source code file for your C++ program into the D2L drop box.
-

Note: *I will eventually provide a testing version of **empl.dat** for you so you can make sure your program works with the format. However, you should create your own test file to get started.*

Output Guidelines

The program should use a menu driven interface. Below is an example of the *approximate* look that the menus should have.

```
                         MAIN MENU

        1)  VIEW an Employee
        2)  EDIT an Employee's Department Number
        3)  LIST all Employees


             4)  END THE PROGRAM


  Please enter your selection from the menu [1-4]  :  __
```

```
                       VIEW EMPLOYEE

        To view an Employee record you will need to
        enter their Employee ID number.

        To return to the MAIN MENU :  enter number   0


  Please enter the Employee ID number:  __
```

```
                            EDIT
               EMPLOYEE DEPARTMENT NUMBER

        To EDIT an Employee Department number you will need to
        enter the Employee's ID number.

        To return to the MAIN MENU :  enter  number   0


  Please enter the Employee ID number:  __
```

```
                              EDIT
              EMPLOYEE DEPARTMENT NUMBER  cont..

        Employee :  {print employee name here}

        Current Department Number :  {print department # here }

        To change the department number you will need to
        enter the new department number.

        To return to the MAIN MENU :  enter  number   0


  Please enter the NEW department number:  __
```

```
                        LIST ALL EMPLOYEES

        You may list all the employees ordered in several different
        ways.  All will be in ascending order.

        1) List ordered by Employee ID number
        2) List ordered by Employee Name {sorry not available}

               3)  Return to MAIN MENU


  Please enter your selection from the menu [1-3]  :  __
```

Output notes:

After listing any output to the screen you should make the user explicitly choose to continue. In that way they have time to look at the output before being returned to the main menu!!

Example: Enter M/m to return the MAIN MENU

Note: you can assume the Binary file will have no more than 100 records so your index can be of that size.
Note: I highly recommend that the data structure you use for your index be an array of records, each with two fields, the Key and the RID. The key holding the empl-id value and the RID holding the record location in the file.