CMPT 201 – Winter 2025, Assignment 3

Due: April 4th @ 11:59
Weight: 8% of your final mark
Work type: Group assignment

Objective

Your assignment is to implement a data processing module and use it to:

- implement an application that can read a data file in .csv format and create a Database (a set of Data Structures) to represent it
- implement an application that can interrogate and modify the data stored in the Database
- create a dashboard which allows users to interact with your implementations
- test your programs extensively, including automated rules for your tests (we want to run the same tests that you have!).

Notes:

This assignment has two milestones which are to be met:

- **Milestone 1: due at 11:59 on March 25th**. Submit on Meskanas a **zip** file named ***your_group_name*_milestone1.tar.gz** that contains two files: the complete header file **DB.h**, and a text file containing details on your plan for working on the assignment. The text file should include work distribution in your team, dates, and issues you anticipate facing.

Specification for the Database module:

The City of Edmonton maintains a dataset of Public Picnic Table Locations. Information about it can be found at https://data.edmonton.ca/Facilities-and-Structures/Public-Picnic-Table-Locations/vk3s-q842. The dataset can be downloaded in several formats. For the purposes of this assignment we will assume a comma separated values (.csv) file format. For more info on .csv see Wikipedia entry https://en.wikipedia.org/wiki/Comma-separated_values

The description of the dataset columns is as follows[1]:

Column Name	Description	Type
Id	Unique Site Identifier for all picnic tables in a given area	Number
Table Type	Overall description of picnic table	Text
Surface Material	Basic material used for the picnic table top	Text
Structural Material	Basic material used to build the picnic table	Text
Street / Avenue	Closest street or Avenue the picnic table is near-by	Text
Neighbourhood Id	Unique Neighbourhood Identifier	Number
Neighbourhood Name	Name of the Neighbourhood the picnic table is found in	Text
Ward	Name of the Ward the picnic table is found in	Text
Latitude	Latitude Coordinate for picnic table	Number
Longitude	Longitude Coordinate for picnic table	Number
Location	General spatial point where this caution or closure is to occur.	Location

A) Design and implement a Database module to read a Public Picnic Table Locations dataset, store it in the specific Database format (your Data Structures), allow specific queries of the Database, allow sorting of the Database. You must make and justify design decisions for your implementation (explain them in the **`design.txt`** file) , based on the following requirements:

- Your module must be able to read the data from a .csv file (similar to the one attached to the assignment) and store it in a series of tables as defined below:
  - **`tableType` Lookup Table** – which contains a code for each possible entry in the Table Type column of the dataset
  - **`surfaceMateria`**l **Lookup Table** – which contains a code and a value for each possible entry in the Surface Material column of the datafile
  - **`structuralMaterial` Lookup Table** – which contains a code value for each possible entry in the Structural Material column of the datafile
  - **`neighbourhood` Lookup Table** – which contains a code and a value for each possible entry in the Neigbourhood Name column of the datafile. The code must be the one used in the Neighbourhood Id field of the record.
  - **`picnicTable` Data Table** - which contains an entry for each record (line) in the original .csv file. Each entry in the table must have the following members:
    - **`tableId`** – a unique id corresponding to the table. This is a new id to be created by your application
    - **`siteId`** – the id of the site of the picnic table – corresponds to the information in the Id column of the record

[1] Description taken from https://data.edmonton.ca/Facilities-and-Structures/Public-Picnic-Table-Locations/vk3s-q842

    - **tableTypeId** – the code of an entry in the **tableType** Lookup Table. This code must correspond to an entry in the lookup table which holds the same information as the Table Type column of the record
    - **surfaceMaterialId** – the code of an entry in the **surfaceMaterial** Lookup Table. This code must correspond to an entry in the lookup table which holds the same information as the Surface Material column of the record
    - **structuralMaterialId** – the code of an entry in the **structuralMaterial** Lookup Table. This code must correspond to an entry in the lookup table which holds the same information as the Structural Material column of the record
    - **streetAvenue** – the information in the Street/Avenue column of the record
    - **neighbhdId** - the information in the Neighbourhood Id column of the record
    - **ward** – **the number** of the ward the table is in; **related** to the ward column of the record
    - **latitude** – the information in the latitude column of the record. This field may be represented as a string in the picnicTable.
    - **longitude** – the information in the longitude column of the record. This field may be represented as a string in the picnicTable.
- The module must implement the following functions:
  - **importDB(file name)** – this function takes the name of a .csv file as parameter and creates and populates the Database with the corresponding data set information.
  - **exportDB(file name)** – this function takes the name of a .csv file as parameter and creates a .csv file containing the information of the Database. NOTE: the exported .csv file must be exactly the same as the original .csv file from which the Database was created.
  - **countEntries(memberName,value)** – this function takes the name of a member of the **picnicTable** entry and a value for that member as parameters and returns the number of entries in the **picnicTable** which have that **value** in the **memberName**.
  - **sortByMember(memberName)** – this function takes the name of a member of the picnicTable entry as an argument and sorts the table in ascending order of the entry values of that member. The function must work for all member types (numeric or text).
  - **editTableEntry(tableID,memberName, newValue)** – this function takes a tableID, the name of a member of the **picnicTable** entry and a value for that member as parameters. It finds the entry which has that **tableID** and changes its **memberName** value to **newValue**.

  memberName in above functions must be spelled as shown in **picnicTable** table description given earlier.
  - **reportByWard()** – this function produces a listing of picnic tables grouped by wards in ascending order.
  - **reportByNeighbourhood()** – this function produces a listing of picnic tables grouped by neigbourhoods in ascending alphabetical order.

- We will test your data processing module implementation under the assumption that it can handle any number of entries (as long as memory is available), but they are all coming from the attached .csv file. **This means that the set of all possible entry values for the lookup tables are indicated by the attached .csv file.**
- You must implement the data processing module, test it appropriately, and then report your effort in a test report.

B) Write a dashboard program which presents a menu allowing the user to choose one of the following:
  - Import Database – ask for file name
  - Export Database – ask for file name
  - Count Entries – ask for member name and value

- Sort By – ask for member name
- Edit Entry – ask for id, member name and value
- Report – ask for Ward or Neighbourhood ID

Your program should not allow any operation to be performed on the Database, until the database is created. For the dashboard to be fully interactive, add an option to exit the program to the menu, and make sure you display output to the user (e.g. value returned by countEntries in sentence form, report by ward or neighbourhood ID).

NOTE: When editing entries, you must maintain your database integrity. For example, if when editing the table type member a value which does not exist in the **`tableType`** lookup table is entered, the **`tableType`** lookup table, must be updated to contain the newly created value.

Description of Assignment:

- Follow **Good Programming Style** document for guidelines on style. **You will use a single global variable (DB, as declared in the header file DB.h) in this assignment. You must not use any other global variables.**
- You must use the C programming language (C99).
- You must create a project on GitHub on which the members of the team can cooperatively work on the assignment.
- You must submit at least four source files, named **makefile**, **`DB.h`**, **`DB.c`** and **`dashboard.c`** (you may also submit additional testing source files or modules if you write them).

Files:

Attached to this assignment you will find the following files:

- **`PicnicTables.csv`** - a picnic table .csv file.
- **`Examples.txt`** –a text file illustrating expected outcomes of each of the required functions.
- **`DB.h`** –a first version of the header file for DB.c. Add struct declarations as indicated in the file and complete the function documentations. This header file should be complete by Milestone 1.

Packaging and Deliverables:

Your tar file should contain a ***your_group_name*** **`_as3`** directory which includes all files related to this assignment. In particular, the ***your_group_name*** **`_as3`** directory should contain:

- a file called **`README`** explaining the contents of each file you have included in the directory.
- makefile, **`make all`** should produce all executables as you described in your user manual
- all source files
- a design document, called **`design.txt`** (or **`design.pdf`** if you prefer a pdf file), that describes the design decisions you made. The design document is supposed to record the important decisions you made about what had to be done (analysis) and how to do it (design). In particular, it should contain the reasons for why you made your decisions. Part of your design document will be the issues list, which identifies the issues that you encountered during the assignment, and how they were resolved.
- a collection of your regression test files, **in a directory called `Tests`**, which also contains a testing report in a text file called **`testing.txt`** (or **`testing.pdf`** if you prefer a pdf file), describing what and how you tested, and why it's sufficient. Don't forget, for this assignment, we want to be able to easily run the (highest level at least) tests that you ran, so be sure to automate their use in your makefile (this can and should include at least one main program that is only for testing your module).

Only include source files and documentation files. Do not include any executable, object or core files.

You are expected to meet normal coding standards.

1. **Packaging:** Makefile must work perfectly, so that all executables are created when `make all` run with a checked out copy of your submission. You should test this yourself. **NO EXECUTABLE FILES SHOULD BE INCLUDED - ONLY SOURCES AND TEXT FILES OR MARKS WILL BE DEDUCTED**
2. **Design:**
   - Were design decisions documented?
   - Was a coherent design approach attempted and how well did it work?
3. **Coding:**
   - Produces reasonable output, even with unreasonable input (i.e., exits with error, warns about problems).
   - Prints out warnings upon detection of problems (i.e., `"Couldn't open file  'blah'!"`).
   - Marks will be deducted for significant coding errors, such as:
     ▪ Not checking the return code where it should be checked.
     ▪ Not matching the specifications pre/post conditions.
     ▪ Failure to document a non-obvious loop. This documentation must convince the reader of the correctness of the loop.
4. **Testing Strategy**:
   - Any programs (source code) used in order to test programs must be included. We expect you to submit the regression tests you are using and explain why you believe them to be a satisfactory test of your program.
5. **Style**:
   - All source files must have header comments giving the usage information and purpose of the program.
   - All additional functions must have function information and purpose documented (put such documentation with the prototype of the function in a header file), as well as having the code documented internally.
   - "Bad" documentation (spurious, misleading or just plain wrong) will be penalized.

Submission: Your compressed tar file will be submitted to Meskanas.

Marking Scheme

The following marking scheme will be used for this assignment:

ITEM	**MARKS**
Milestone 1	**6**
Packaging: module has correct structure(README) , tarfile extracts properly, compiles and runs	**4**
makefile; make all, testing targets, etc.	**10**
DB module passes our tests	**40**
Input Validation and Error Handling	**5**
Correct dashboard program	**8**
Module testing strategy, test documentation	**12**
Design document, design decisions justifications	**5**
Module and program documentation, coding style	**5**
Program design, good program structure, use of multiple files, etc	**3**
GitHub project created and used properly	**2**
TOTAL	**100**