Programming Assignment #5A: Managing a vector of *DateType* objects

Overview:

Imagine that we want to manage a list of birthdays of your family and friends so that we can conveniently sort birthdays, search for birthdays within a given range, store the birthdays in a file, and read them from a file.

To accomplish the task, we'll use a *vector<DateType>* object as a container in the main memory to store and manage the birthday information and use various operators for the *DateType* class implemented in programming #2 for input, output, and comparisons of dates.

In addition, we'll also adapt the implementation of *mergeTwoSortedVectors* and *mergeSort* functions implemented in programming #3 for sorting the birthday information as a *vector<DateType>* object.

****

Step 1. Revise the functions *mergeTwoSortedVectors* and *mergeSort* in Programming #3 to provide the capability of sorting *DateType* objects instead of *double* values:

- First, make a copy of your entire programming project for **programming #3**. Rename the .cpp file that contains your main function as **dateDB.cpp.** For Visual C++, you should then open up the project and add both of them into the project go to Project /Add Existing Item to do so. For Visual C++, under the solution view you can directly rename the file.
- Add a copy of **DateType.h** and a copy of **DateType.cpp** from programming #2 into the source code subfolder in this project. For Visual C++, you can add both of them into the project by going to "Project /Add Existing Item" to do so.
- Add the line *#include "DateType.h "* and the line *#include <vector>* in the beginning of the .cpp file containing the main function . Modify the prototype

  *bool mergeTwoSortedVectors(vector<**double**> & vecA, vector<double> & vecB, vector<double> & vecC)*

  to *bool mergeTwoSortedVectors*

  *(vector<**DateType**> & vecA, vector< **DateType** > & vecB, vector< **DateType** > & vecC)*

and modify the implementation so that it can merge two sorted vectors of *DateType* objects.

- Modify the prototype of

  bool mergeSort(vector<double> & vecToSort) **to** *bool mergeSort(vector<DateType> & vecToSort)*

  and modify the implementation so that it can sort a vector of *DateType* objects.

- **Testing**: **Modify the main function so that the user can enter dates (instead of *double* values) to make sure the newly modified functions can work well for merging and sorting dates respectively**.

Here is an an example executable (zipped) of what you should have by the end of Step 1 for testing. Note that the testing code in the main function in Step 1 is not needed for in the final version for submission. See Step 2 below about this.

****

Step 2. Completely rewrite the main function entirely to implement in dateDB.cpp to provide service:

Overview: After Step 1, your main function in **dateDB.cpp** should be completely rewritten now to provide services to read dates from the keyboard or files, save the dates into files, search for dates within a given range of dates, and sort dates in order. To accomplish these purposes, in the main function we'll use a *vector<DateType>* object as a container in the main memory to store and manage the date information by using various operators and member functions of the *DateType* class. See more details below.

Key data structures and variables:

- To dynamically store the information of dates, you should declare in your main function " *vector<DateType> dateDB;* " to create a local vector for storing *DateType* objects. You should also declare in your main function local *DateType* objects " *DateType date, dBegin, dEnd;* " for storing temporary date information.

- Add the line *#include <string>* in the beginning of the .cpp file for including the string processing support. In the main function, you should declare a string variable " *string filename;* " for storing the file name given by the user.
- Add the line *#include <fstream>* in the beginning of the .cpp file for file I/O support. You should declare also two file handles " *ifstream fin;* " and " *ofstream fout;* " for file input/output purposes respectively.
- You should also declare in your main function local integer variable " *int numDateRecords;* " for temporarily storing the information of the number of date records involved in the file I/O.

Service menu: The main function should set up a loop that would repeatedly display a menu to prompt the user to choose one of the following services:

- ***K***: to read a date manually ***K***eyed in by the user and store it (i.e. *push_back*) in the end of the vector *dateDB*,
- ***B***: to **clear** the vector *dateDB* into an empty vector first and then **read** a ***B****atch* ***of*** dates from a file specified by the users into the vector *dateDB*,
- ***D***: to ***D***isplay the dates currently stored in the vector *dateDB* to the screen,
- ***M***: to ***M***odify one of the dates by reading a new date from the keyboard to overwrite the contents of an existing *DateType* object (specified by the user) in the vector *dateDB*,
- ***W***: to ***W***rite the dates currently stored in the vector *dateDB* into a file specified by the user,
- ***F***: to ***F***ind and display dates within a range specified by the user,
- ***S***: to ***S***ort the dates currently stored in the vector *dateDB* in order,
- **Q**: to ***Q***uit the program.

Implementation of the services: On each iteration of that loop, the program should read the user's choice and use a *switch* statement to do things according to the choice:

- If the choice is '**K**': the program should ask the local *DateType* object *date* to read in a new date (i.e. " cin >> *date;* ") and then add the date into the end of the vector *dateDB* (i.e. " *dateDB.push_back(date);* ").

- If the choice is '**B**': the program should ask the user to provide the name of an input file (e.g. *dates.txt*), and read the file name into the string variable *filename (i.e.* " *cin >> filename;* "). The program should then open that file through the input file object *fin (i.e.* " *fin.open(filename);* ") and read the first integer from the file into *numDateRecords (i.e.* " *fin >> numDateRecords;* "), and the program should check to make sure this number is non-negative. If it is positive, the program should clear *dateDB (i.e.* " *dateDB.clear( );* ") to an empty vector, and then set up a loop that iterates for *numDateRecords* iterations to repeatedly read in a date from the file into the local *DateType* object *date* (i.e. " *fin >> date;* ") and then add the date into the end of the vector *dateDB* (i.e. " *dateDB.push_back(date);* "). After the loop is finished, the program should close that file *(i.e.* " *fin.close( );* ").

- If the choice is '**D**': the program should set up a loop to display each element *dateDB[i]* in the vector *dateDB (i.e.* " *cout << dateDB[i] ;* ").

- If the choice is '**M**': the program should ask the user the index *i* of the date in *dateDB[i]* he/she wants to modify, read the index as an integer from the user, check to make sure *i* is non-negative and less than *dateDB.size( )*, and if so read the new date into *dateDB[i]* (i.e. " *cin >> dateDB[i] ;* ").

- If the choice is '**W**': the program should ask the user to provide the name of an output file (e.g. *dates.txt*), and read the file name into the string variable *filename (i.e.* " *cin >> filename;* "). The program should then open that file through the output file object *fout (i.e.* " *fout.open(filename);* ") and first write the size of *dateDB* into the file *(i.e.* " *fout << dateDB.size( ) << endl;* "). The program should then set up a loop that iterates for *dateDB.size( )* iterations to repeatedly write each element *dateDB[i]* into the file (i.e. " *fout*

<< *dateDB[i])* << *endl;* ”). After the loop is finished, the program should close that file *(i.e.* ” *fout.close( );* ”).

- If the choice is ‘**F**’, the program should ask the user to provide the beginning date and the ending date of the date range *(i.e.* ” *cin >> dBegin;* ” and ” *cin >> dEnd;* ”), and then the program should set up a loop to check each element *dateDB[i]* in the vector *dateDB* and display the contents of *dateDB[i]* if *dateDB[i]* is within the range *(i.e.* if ” dBegin<= *dateDB[i] && dateDB[i] <= dEnd* ” is true).

- If the choice is ‘**S**’, the program should call the revised mergeSort function to sort the dates in order *(i.e.* ” *mergeSort(dateDB);* ”).

- If the choice is ‘**Q**’, the program will output a thank-you message, exit the loop, and end the program.

****

Step 3. Tesing and Submission: When you are sure that all the services in the menu provided by the main function are working, you are done. Fill out this Test3_self-evaluation report and submit (i) the entire program folder containing your source code files (.cpp and .h files) together with (ii) the self-evaluation report as **a single zip file** under Canvas.