Programming Assignment #5B: Managing a linked list of *DateType* objects

Overview:

Just like in Programming Assignment #5A, 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. However, this time we'll use a *linked list* as a container in the main memory to store and manage *DateType* objects and use various operators for the *DateType* class implemented in programming #2 for input, output, and comparisons of dates. **Note that we do not offer the soring option in Programming Assignment#5B since we'll maintain the linked list as a sorted list**.

****

Step 1. Modify the implementation of the linked list class for storing integers into a linked list class for storing dates:

- **Download**: Download examine with this sample linked list C++ project regarding the implementation of a simple linked list class that allows a linked list object to stores integers in the *ListNode* structures and maintain these *ListNode* structures as a linked list.
- **Testing**: Run the program and carefully examine the main function in **main.cpp**, the declaration of the *List* class in **LinkedList.h**, and the implementation of the *List* class in **LinkedList.cpp** to see how a *List* object can insert or delete an integer from list, display the integers in the whole list, display the integers between two given dates, read (or write) a batch of integers from (or into) a file (or from the keyboard input) into the linked list.
- **Add the support of DateType class:**
    1. Copy your own **DateType.h** and **DateType.cpp** that you have implemented for programming #2 into the project folder and add them into your project.
    2. Add **#include "DateType.h"** in the beginning of **LinkedList.h** to inform the compiler of the declaration of the **DateType** class before the compiler processes the declaration of the new *List* class you are going to implement.
- **Revise the declaration of the *List* class**: Modify the declaration of the *ListNode* structure and the prototypes of the member functions of the *List* class in **LinkedList.h** to have a new *List* class that allows a linked list object to store *DateType* objects (instead of *int* value) in the *value* field of the *ListNode* structure and maintain a collection of such *ListNode* structures as a linked list.
- **Implement the new *List* class**: Modify the implementation (i.e. the definition) of the member functions of the *List* class in **LinkedList.cpp** to enable a *List* object of your new *List* class to stores *DateType* objects in the new *ListNode* structures and maintain these *ListNode* structures as a linked list**. Note that** many local variables used in these member functions now need to be *DateType* objects instead of *int* variables.
- **Testing**: Modify the *main* function in **main.cpp** accordingly to extensively test whether a *List* object of the new *List* class you created above can correctly manage a collection of *DateType* objects as a linked list. Use the code in the *main* function

to call member functions of the new *List* class to make sure a *List* object can insert or delete a date from list, display the dates, display the dates between two given dates, read (or write) a batch of dates from (or into) a file (or from the keyboard input) into the linked list.

****

Step 2. Rewrite the main function in main.cpp to provide the services:

Overview: As in Programming Assignment #5A, you should now totally rewrite the main function in **main.cpp** to provide services to read dates from the keyboard or files, save the dates into files, and search for dates within a given range of dates. You can reuse much of the code framework you have in the main function for Programming Assignment #5A to accomplish these purposes.

However, this time in the main function we'll use an object of the new *List* class 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. It is basically almost the same as what you did in Programming Assignment #5A, except for the following things:

- We'll maintain the list as a sorted linked list by always calling the *InserInOrder* method to insert a new *DateType* object into the linked list. **Therefore we do not offer the soring option in Programming Assignment#5B.**
- You have to deal with each option by calling the member functions in the new *List* class. See the details in the section of Implementation of the services below.

Key data structures:

- To dynamically store the information of dates, you should declare in your main function " *List dateDB;* " to create a local linked list object for managing the information of *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 <fstream>* in the beginning of the .cpp file for file I/O support. In the main function, you should declare a string object " *string filename;* " for storing the file name given by the user. You should declare 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 in one more date manually ***K***eyed in by the user and insert it in the right place in the linked list to maintain a sorted linked list of dates.

- ***B***: to **clear** the list *dateDB* into an empty list first and then **read** a ***Batch of*** dates from a file specified by the users into the list *dateDB,*
- ***D***: to ***D***isplay the dates currently stored in the list *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 linked list *dateDB,*
- ***W***: to ***W***rite the dates currently stored in the list *dateDB* into a file specified by the user,
- ***F***: to ***F***ind and display dates within a range specified by the user,
- **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 insert the date into the right place within the linked list *dateDB*, i.e. just call *dateDB.InserInOrder(date)* to insert it in the right place in the linked list.
- 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 object *filename (i.e. " cin >> filename; "*). The program should then open that file through the input file object fin (i.e. " fin.open(filename); ") and have dateDB read all the dates from the file by the statement " fin >> dateDB; " . After that, the program should close that file *(i.e. " fin.close( ); "*).
- If the choice is '**D**': the program should call *dateDB.Display( )* to display each date store in the linked list *dateDB.*
- If the choice is '**M**': the program should ask the user to enter the date to modify and the read the date (i.e. " *cin >> date;* "), call *dateDB.Remove(date)* to delete the date first, and then ask the user to enter the new date to replace it and read the date (i.e. " *cin >> date;* "), and finally call *dateDB.InserInOrder(date)* to insert it in the right place in the linked list.
- If the choice is '**W**': the program should ask the user to provide the name (less than 20 characters) of an input file (e.g. *dates.txt*), and read the file name into the string object *filename (i.e. " cin >> filename; "*). The program should then open that file through the output file object *fout (i.e. " fin.open(filename); "*) and have *dateDB* write the number of dates and each individual date into the file by the statement " *fout << dateDB;* " . After that, 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 have *dateDB* display all the dates in theat range by calling *dateDB.FindAndDisplay(dBegin, dEnd).* See the note below.
- **Note that** *void FindAndDisplay(DateType dBegin, DateType dEnd)* is a member function in the *List* class: This member function sets up a loop to check each element in the linked list and display the dates stored in the nodes of the linked list which are within the range of the two date objects *dBegin* and *dEnd*.
- If the choice is '**Q**', the program will output a thank-you message, exit the loop, and end the program.