CS 2401 – Spring 2023

Project 3 – Linked Lists

Due: 11:59 PM Wednesday, March 8th

You were just hired by a pharmacy to help them manage their vaccine appointments. The pharmacy keeps running out of the flu vaccine (it is flu season after all), so they need you to write some software to keep track of customers waiting for a vaccine. **The line must be kept in order by the time that the person called in to ask for an appointment.**

To get us started on this project, the *Date* class that we used in the last two projects has been altered so that you can create a date object set to today using the computer's internal clock. There is also a function that allows us to know someone's age by sending a "today" object to another *Date* object. The return value of this function is an integer.

In the same file (to keep the number of files that we are working with smaller) there is a new class called *DateTime*, which records the hour and minute when something happened as well as a date.

Using these two, I have written a class called *Appointment*, which records the name of a person (a string), their birthday (a *Date*) and the date/time when they called in to ask for an appointment (a *DateTime*). The overloaded comparison operators for the *Appointment* class are based on the call-in time, so if appoint1 < appoint2 it means that appoint1 called in before appoint2, disregarding name and birthday.

You should begin the project by downloading:

datetime.h
datetime.cc
appointment.h
appointment.cc
node.h (has the appointment as its data type)
main.cc
inline.txt

Your task is to create is to create the header and implementation files for the WaitList class. This class will store a list of people who have called in for appointments, meaning that they are waiting for an appointment to become available. This list is stored in the form of a **linked list of nodes** (I have given you the node class you are to use). The list is **always** kept in order by the call-in time, but the people who put the things into the list may not be recording them in that order. This means that when a new appointment is recorded you will need to search through the list to put it in the right place. To facilitate this, a**ll comparison operators are already overloaded for the Appointment class**.

The WaitList class will offer the application programmer the following options:

1. Add a new Appointment request to the list. (It will be inserted in order.)
2. View the entire list of Appointments, in the order they are stored.
3. Remove a person's appointment because they have found an appointment at another location. This will use the find function, and both will take a string which is the name of the patient.
4. Learn the number of people waiting in line.
5. See the wait time for the person waiting the longest. (This person should be at the front of the line.)
6. See the average wait time for everyone in the list.
7. Identify the oldest person in the list.
8. Identify the youngest person in the list.
9. See the average age of those waiting.

The main that I have given you has a menu for all these options, and you should be able discern the names and parameters for the required functions from the calls that you see in the main.

You will also need to have a load() and a save() that saves just the information about the appointments (do not save the size of the list). I have provided a sample file for this purpose. The sample data file is in the format: patient name, patient bday, date patient called, time patient called.

Since linked lists are built using dynamic memory, your class will be "holding" dynamic memory and that means that you will need the implementation of **the Big 3**, and I have built a "hidden" menu option that will let me (and you) check your copy constructor.

For this project you should submit the w*aitlist.h* and w*aitlist.cc* that you have created.