CS 133, Spring 2023
Programming Project #3: Tag (20 points)
Due Tuesday, May 2nd, 2023, 11:59 PM

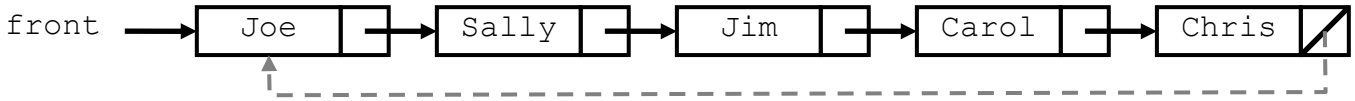
Thanks to Stuart Reges and Marty Stepp for parts of this assignment

This program focuses on implementing a linked list. Turn in a files named `TagManager.h` and `TagManager.cpp` from the Project section of the course web site. You will need `tagStarterProject.zip` from the Project section of the course web site which contains the support files `TagNode.h`, `TagNode.cpp`, `tagMain.cpp`, and `names.txt`.

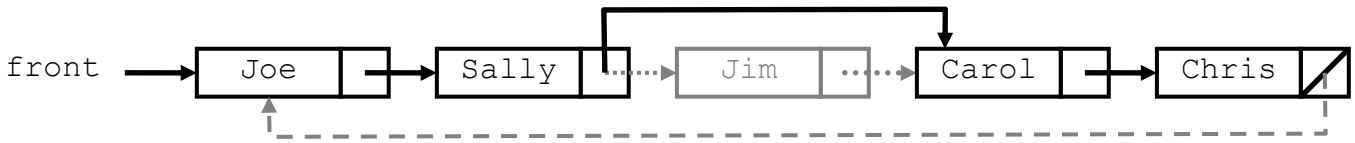
Program Description:

You probably played several different versions of tag in elementary school. In this version, each person playing has a particular target that he/she is trying to tag. One of the things that makes the game more interesting to play in real life is that initially each person knows only who they are trying to tag; they don't know who is trying to tag them, nor do they know whom the other people are trying to tag.

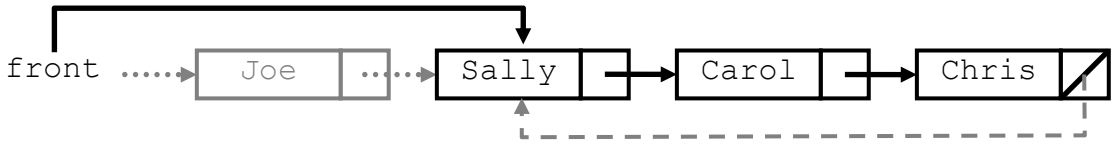
The game of tag is played as follows: You start out with a group of people who want to play the game. For example, let's say that there are five people playing named Carol, Chris, Jim, Joe, Sally. A circular chain of tag targets (called the "tag ring" in this program) is randomly established. For example, we might decide Joe should try to tag Sally, Sally should try to tag Jim, Jim should try to tag Carol, Carol should try to tag Chris, and Chris should try to tag Joe. (In the actual linked list that implements this tag ring, Chris's `next` pointer would be `null`. But conceptually we can think of it as though the next person after Chris is Joe, the front person in the list.) Here is a picture of this "tag ring":



When someone is tagged, the links need to be changed to "skip" that person. For example, suppose that Sally tags Jim. Sally needs a new person to try to tag, so we give her Jim's target: Carol. The tag ring becomes:



If the first person in the tag ring is tagged, the front of the list must adjust. If Chris tags Joe, the list becomes:



You will write a class `TagManager` that keeps track of who is trying to tag whom and the history of who tagged whom. You will maintain two linked lists: a list of people currently in the game (the "tag ring") and a list of those who are out (the "out list"). As people are tagged, you will move them from the tag ring to the out list by rearranging links between nodes. The game ends when only one node remains in the tag ring, representing the winner.

A client program called `TagMain` has been written for you. It reads a file of names, shuffles the names, and constructs an object of your class `TagManager`. This main program then asks the user for the names of each player to record as tagged until there is just one player left in the game (at which point the game is over and the last remaining player wins). `TagMain` calls member functions of the `TagManager` class to carry out the tasks involved in administering the game.

Sample Log of Execution (user input underlined):

Your program should exactly reproduce the format and general behavior demonstrated in this log, although you may not exactly recreate this scenario because of the shuffling of the names that `tagMain` performs.

```
Current tag ring:
  Erica Kane is trying to tag Ruth Martin
  Ruth Martin is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Bobby Warner
  Bobby Warner is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current out list:

next person tagged? Ruth Martin

Current tag ring:
  Erica Kane is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Bobby Warner
  Bobby Warner is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current out list:
  Ruth Martin was tagged by Erica Kane

next person tagged? Ruth Martin
Ruth Martin is already out.

Current tag ring:
  Erica Kane is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Bobby Warner
  Bobby Warner is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current out list:
  Ruth Martin was tagged by Erica Kane

next person tagged? bobby warner

Current tag ring:
  Erica Kane is trying to tag Jackson Montgomery
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Erica Kane
Current out list:
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person tagged? ERICa kaNE

Current tag ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Anita Santos
  Anita Santos is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Jackson Montgomery
Current out list:
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person tagged? ANITA SANTOS
```

(continued)

```
Current tag ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Tad Martin
  Tad Martin is trying to tag Phoebe Wallingford
  Phoebe Wallingford is trying to tag Jackson Montgomery
Current out list:
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person tagged? phoebe wallingford

Current tag ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Tad Martin
  Tad Martin is trying to tag Jackson Montgomery
Current out list:
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person tagged? Barack Obama
Unknown person.

Current tag ring:
  Jackson Montgomery is trying to tag Joe Martin
  Joe Martin is trying to tag Tad Martin
  Tad Martin is trying to tag Jackson Montgomery
Current out list:
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person tagged? Joe Martin

Current tag ring:
  Jackson Montgomery is trying to tag Tad Martin
  Tad Martin is trying to tag Jackson Montgomery
Current out list:
  Joe Martin was tagged by Jackson Montgomery
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane

next person tagged? jackson montgomery

Game was won by Tad Martin
Final out list is as follows:
  Jackson Montgomery was tagged by Tad Martin
  Joe Martin was tagged by Jackson Montgomery
  Phoebe Wallingford was tagged by Tad Martin
  Anita Santos was tagged by Joe Martin
  Erica Kane was tagged by Phoebe Wallingford
  Bobby Warner was tagged by Jackson Montgomery
  Ruth Martin was tagged by Erica Kane
```

Implementation Details:

You must use our node class `TagNode` (provided in the starter project) which has the following header contents:

```
struct TagNode {
    string name;          // this person's name
    string tagger;        // name of who tagged this person (null if in the game)
    TagNode* next;        // next node in the list

    TagNode(string name);
    TagNode(string name, TagNode* next);
}
```

For this assignment we are going to specify exactly what member variables you can have in your `TagManager` class. **You must have exactly the following two member variables**; you are not allowed to have any others:

- a pointer to the front node of the tag ring
- a pointer to the front node of the out list (`nullptr` if empty)

Implementation Details (continued):

Your class should have the following public member functions.

`TagManager(vector<string> names)`

In this constructor you should initialize a new tag manager for the given list of people. Your constructor should not save the list parameter itself as a member variable, nor modify the list; instead, it should build your own tag ring of linked nodes that contains these names in the same order. For example, if the list contains `["John", "Sally", "Fred"]`, the initial tag ring should represent that `John` is trying to tag `Sally` who is trying to tag `Fred` who is trying to tag `John` (in that order). You may assume that the names are non-empty strings and that there are no duplicates.

You should throw a `string` exception if the list has a size of 0.

`void printTagRing()`

In this member function you should print the names of the people in the tag ring, one per line, indented by two spaces, as "**name** `is trying to tag` **name**". The behavior is unspecified if the game is over. For the names on the first page, the initial output is:

```
  Joe is trying to tag Sally
  Sally is trying to tag Jim
  Jim is trying to tag Carol
  Carol is trying to tag Chris
  Chris is trying to tag Joe
```

`void printOutList()`

In this member function you should print the names of the people in the out list, one per line, with each line indented by two spaces, with output of the form "**name** `was tagged by` **name**". It should print the names in the opposite of the order in which they were tagged (most recently tagged first, then next more recently tagged, and so on). It should produce no output if the out list is empty. For example, from the previous names, if `Jim` is tagged, then `Chris`, then `Carol`, the output is:

```
  Carol was tagged by Sally
  Chris was tagged by Carol
  Jim was tagged by Sally
```

`bool tagRingContains(string name)`

In this member function you should return `true` if the given name is in the current tag ring and `false` otherwise. It should ignore case in comparing names; for example, if passed `"salLY"`, it should match a node with a name of `"Sally"`.

`bool outListContains(string name)`

In this member function you should return `true` if the given name is in the current out list and `false` otherwise. It should ignore case in comparing names; for example, if passed `"CaRoL"`, it should match a node with a name of `"Carol"`.

`bool isGameOver()`

In this member function you should return `true` if the game is over (if the tag ring has just one person) and `false` otherwise.

`string winner()`

In this member function you should return the name of the winner of the game, or an empty string if the game is not over.

`void tag(string name)`

In this member function you should record the tagging of the person with the given name, transferring the person from the tag ring to the front of the out list. This operation should not change the relative order of the tag ring (i.e., the links of who is trying to tag whom should stay the same other than the person who is being tagged/removed). Ignore case in comparing names. A node remembers who tagged the person in its `tagger` member variable. It is your code's responsibility to set that member variable's value. **You may not change the `name` member variable of any nodes**.

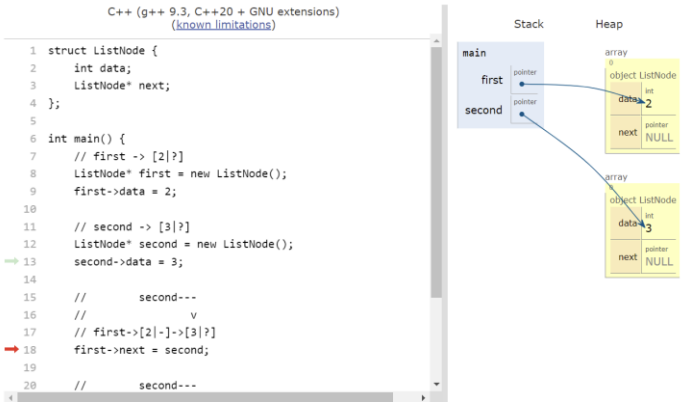
You should throw a `string` exception of `"game over"` if the game is over, or a string exception of `"not in ring"` if the given name is not part of the tag ring (if both of these conditions are true, the `"game over"` takes precedence).

The `tag` member function is the hardest one, so write it last. Use the debugger, C++ Tutor and `cout` statements liberally to debug problems in your code. You will likely have a lot of exceptions, errors, infinite loops, etc. and will have a very hard time tracking them down unless you are comfortable with debugging techniques.

For full credit, every member function's runtime should be at worst O(*N*), where *N* is the number of people in your linked lists.

C++ Tutor:

In the C++ Tutor (http://pythontutor.com/cpp.html) you can use a structure viewer to see what your list looks like by writing a simple main program that constructs a list and calls the function you want to test and then stepping through your program. We highly recommend using this to debug your program, especially if nodes are disappearing and you can't figure out why.



Other Constraints and Tips:

This is meant to be an exercise in linked list manipulation. As a result, you must adhere to the following rules:

- You must use our `TagNode` class for your lists. You are not allowed to modify it.
- You may not construct any arrays, `vector`s, `list`s, `stack`s, `queue`s, `set`s, `map`s, or other data structures; you must use linked nodes. You may not modify the list of `string`s passed to your constructor.
- If there are *N* names in the list of `string`s passed to your constructor, you should create exactly *N* new `TagNode` objects in your constructor. As people are tagged, you have to **move** their node from the tag ring to the out list by changing pointers, without creating any new node objects or resetting the `name` stored in any node.

Your constructor will create the initial tag ring of nodes, and then your class may not create any more nodes for the rest of the program. You are allowed to declare as many local variables of type `TagNode*` (like `current` from lecture) as you like. `TagNode*` variables are not node objects and therefore don't count against the limit of *n* nodes.

You should write some of your own testing code. `TagMain` requires every member function to be written in order to compile, and it never generates any of the exceptions you have to handle, so it is not exhaustive. You may share your testing code with others on Discord so long as the code does not give away the details of your program solution.

A word of caution: Some students try to store the tag ring in a "circular" linked list, with the list's final element storing a `next` pointer back to the front element. But we discourage you from implementing the program in this way; we strongly suggest that you follow the normal convention of having `nullptr` in the `next` field of the last node. Most people find it difficult to work with a circular list when they are learning to program; it is easy to end up with infinite loops or other bugs. There is no need to use a circular list because you can always get back to the front via the member variables of your `TagManager`. If you feel strongly that you want to use a circular list, you are allowed to do so, but it is likely to make the program harder to write.

Style Guidelines and Grading:

Part of your grade will come from appropriately utilizing linked lists and nodes as described previously. Redundancy is another major grading focus; some member functions are very similar, and you should avoid repeated logic as much as possible.

You should follow good general style guidelines such as: making member variables and extra helper functions `private` and avoiding unnecessary member variables; appropriately using control structures like loops and `if`/`else` statements; properly using indentation, good variable names and proper types; and not having any lines of code longer than 100 characters. **You may not use recursion.**

Comment your code descriptively in your own words in your header file at the top of your class and each member function. Comment descriptively in your own words in your cpp file at the top of your file and on complex sections of your code. Comments should explain each member function's behavior, parameters, return, pre/post-conditions, and exceptions.