DO NOT OPEN EXAM UNTIL INSTRUCTED TO BEGIN

Name: _____ **UIN**: _____ **Instructor**: _____

- **This exam has X questions.**
- **We remove staples and scan only the front pages of exams; therefore,**
  - **Do not write on the back of exam pages.** *Request blank paper if you need it. Write your name and the question number on it and attach to the back of the exam.*
  - **Leave a quarter inch margin around the edges of the page.** *Otherwise, what you write may get chopped off during scanning.*
- **This is a closed book, closed note exam.** *Do not use any notes, books, or computing technology including smart watches. Do not confer with any other person.*
- **Partial credit will be given.** *Do things to make your thinking and process visible, like showing the values of variables as they change.*
- **Grading will be based on correctness, clarity, and neatness.**
- **Suggestion: Read the entire exam before you begin work on any problem.** *Budget your time wisely, according to point distribution.*
- **Make sure you have an ID.** *Your exam will not be graded until identity is confirmed.*
- **This is a 50-minute exam. When the proctor states that the exam period has concluded, stop writing immediately; you will receive a score of *zero* if you continue writing.**
  - **Make sure you've written your name on each page prior to the end of the exam; you will not be allowed to do this once time has expired.**

Please sign to acknowledge the statements above and affirm the Texas A&M University academic integrity statement below:
"On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work. In particular, I certify that I have not received or given any assistance that is contrary to the letter or the spirit of the guidelines for this exam."

Signature: _____

Topics covered: Classes, Rule of Three (destructor, copy constructor, copy assignment), Linked lists

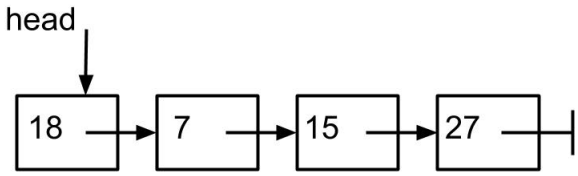
Disclaimer:

The number of questions on this practice exam is not an indicative of the number of questions on the exam. The expected difficulty of the questions on this practice exam is greater than or equal to the expected difficulty of the questions on the exam.

Practice Questions

1. Write a member function of the class `LinkedList` (a simply-linked list, with a pointer `head` to its first element) that returns the average of the elements in the list. You can assume that the list has at least one element.

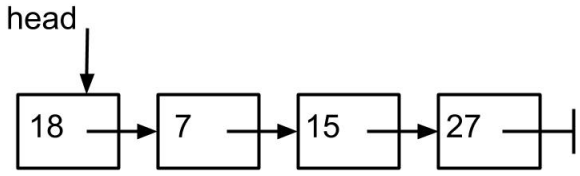
   Example: For the list

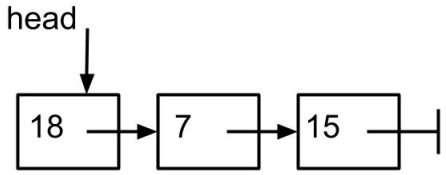
   the function should return 16.75.

2. Write a member function of the class `LinkedList` that removes from the list all occurrences of the maximum element in the list (observe that the list can be and can become empty). Examples:

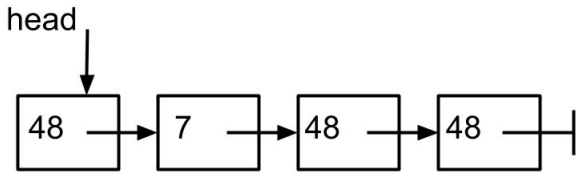
   - The list

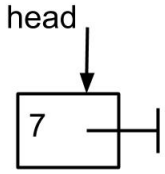
will become



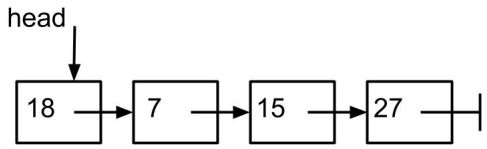
- The list



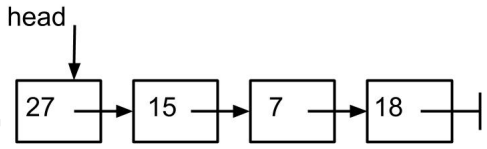
will become



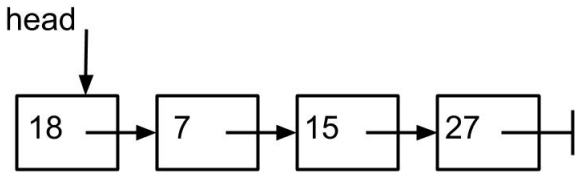
3. Write a member function of the class `LinkedList` that reverses the list. For example, the list



will become



4. Write a member function of the class `LinkedList` that returns a pointer to the element in the middle of the list (if the list has n elements, it should return a pointer to the $\lfloor \frac{n}{2} \rfloor$-th element). For example, for the list



the function should return a pointer to the node that contains 7.

5. You have been asked to help with the development of an application to keep track of highways. Information is loaded from files and added to your transportation application. Your application needs to be able to list the information in your system: list cities and highways, list the highways in a given state, and list the cities crossed by a given highway. The file with information about cities has, for each city, its id, name, state, and population. The file with highway information has, for each highway, its id, the number of cities it crosses, and the id of each of these cities.

   You are asked to model the problem, i.e., to identify the classes you will use to build your application. For each class, provided its header file `.h` that declares the class and its data and function members. You do not need to implement any of the functions.

6. You have been asked to design a class for a TicTacToe game and supporting functions. The game will have a resizable board, so you plan to put the array of char on the heap.

   Design and implement the classes and functions needed for the following main function work. Since the class places memory on the heap, you'll need to implement the rule of three.

```
1  #include <iostream>
2  #include "TicTacToe.h"
3  using namespace std;
4
5  int main() {
6    char again = 'y';
7    while (tolower(again) == 'y') {
8        int numRows = 0;
9        int turnNum = 0;
10       /* returns a value from the user and
11        * ensures that it is between 3 and 9 */
12       numRows = getRows();
13
14       /* TicTacToe class, the constructor creates a
15        * numRowsXnumRows array of char on the heap
16        * which will have spaces and set to 'X' and 'O'
17        * as the game is played */
18       TicTacToe game(numRows);
```

```
19        do { // play game
20            /* prints blanks for empty cells and uses
21             * capital 'X' and 'O' otherwise */
22            game.printBoard();
23            turnNum++;
24            char curLetter;
25            if (turnNum % 2 == 0) {
26                curLetter = 'O';
27            }
28            else {
29                curLetter = 'X';
30            }
31            int row, col;
32            /* getMove sets row and col to values from user
33             * and ensures valid, available location on the
34             * board and uses curLetter to  indicate
35             * who is entering move */
36            game.getMove(row, col, curLetter);
37            // curLetter must be 'X' or 'O' or 'x' or 'o'
38            game.makeMove(curLetter, row, col);
39
40            // while no winner and not all places are full
41        } while (!game.hasWinner()
42                 && (turnNum != game.getWidth()*game.getWidth()));
43
44        game.printBoard();
45        char winner = game.getWinner();
46        /* getWinnerreturns the winning letter or
47         * a 'C' if it is cat (i.e. a tie) */
48        if (winner == 'C') {
49            cout << "Cat!" << endl;
50        }
51        else {
52            cout << winner << " wins!" << endl;
53        }
54        cout << "Do you want to play again? " << endl;
55        cin >> again;
56    } // end of while statement
57 }
```

Typical run:

```
Enter a value between 3 and 9: 3
   |   |
-----------
   |   |
-----------
   |   |
Enter row, col for X: 2 2
   |   |
-----------
   | X |
-----------
   |   |
Enter row, col for O: 3 3
   |   |
-----------
   | X |
-----------
   |   | O
Enter row, col for X: 1 1
 X |   |
-----------
   | X |
-----------
   |   | O
Enter row, col for O: 2 3
 X |   |
-----------
   | X | O
-----------
   |   | O
Enter row, col for X: 1 3
 X |   | X
-----------
   | X | O
-----------
   |   | O
Enter row, col for O: 3 1
 X |   | X
-----------
   | X | O
-----------
 O |   | O
Enter row, col for X: 1 2
```

```
 X | X | X
-----------
   | X | O
-----------
 O |   | O
X wins!
Do you want to play again?
y
Enter a value between 3 and 9: 3

   |   |
-----------
   |   |
-----------
   |   |
Enter row, col for X: 1 1
 X |   |
-----------
   |   |
-----------
   |   |
Enter row, col for O: 3 3
 X |   |
-----------
   |   |
-----------
   |   | O
Enter row, col for X: 1 3}
 X |   | X
-----------
   |   |
-----------
   |   | O
Enter row, col for O: 1 2
 X | O | X
-----------
   |   |
-----------
   |   | O
Enter row, col for X: 2 2
 X | O | X
-----------
   | X |
-----------
   |   | O
```

```
Enter row, col for O: 3 1
 X | O | X
-----------
   | X |
-----------
 O |   | O
Enter row, col for X: 3 2
 X | O | X
-----------
   | X |
-----------
 O | X | O
Enter row, col for O: 2 2
Position taken
Enter row, col for O: 2 1
 X | O | X
-----------
 O | X |
-----------
 O | X | O
Enter row, col for X: 2 3
 X | O | X
-----------
 O | X | X
-----------
 O | X | O
Cat!
Do you want to play again?
n
```


Additional Challenges

In case you are interested and have time and energy for a couple of more challenging questions that you can discuss with your friends who are majoring in computing, the following questions appeared in job interviews of high-profile tech companies:

- Solve Question 3 (reverse a linked list) by updating the `next` pointers of the nodes as you traverse the list. (Instead of the common solution of building the reversed list by adding the elements in reverse order to a new list, one-by-one.)
- Solve Question 4 (find middle of linked list) without counting the number of elements in the list.