Homework 3

Out: 9.29.20
Due: 10.7.20

1. [Recurrences, 28 points]
Give the tightest asymptotic bounds you can for the following recurrences and provide a short explanation for your solution. You may assume that T(1) = constant.

a. $T(n) = 3T(n/9) + \sqrt{n}$
b. $T(n) = 3T(n/9) + 5$
c. $T(n) = 3T(n/9) + n^3$
d. $T(n) = 4T(n/2) + n^2 \log n$
e. $T(n) = 7T(n/2) + n^2$
f. $T(n) = T(n-1) + T(n/2) + n$
g. $T(n) = \begin{cases} 2 & if\ n = 2 \\ 2T\left(\frac{n}{2}\right) + n & if\ n = 2^k, for\ k > 1 \end{cases}$

The solution for the above recurrence is T(n) = n lg n. Prove by induction that this indeed is the solution.

2. [Recurrences, 20 points]
For each of the following functions, provide:
- A recurrence, or other expression, T(n) that describes the worst-case runtime of the function in terms of n as provided (i.e. without any optimizations)
- The tightest asymptotic upper and lower bounds you can find for T(n) (or the runtime)

a.
```
int A(int n) {
  int sum=0;
  for (i=0; i<n; i++)
    for (j=0; j<n; j++)
      sum += i * j % 5;
  return sum;
}
```

b.
```
int B(int n) {
  if (n == n/2)
    return 1;
  else
    return 12*B(n/20);
}
```

c.
```
int C(int n)
{
   if (n == 0) return 1;
   if (C(n/2) >= 5)
         return C(n/2) + 5;
   else
         return 5;
}
```

d.
```
int D(int n) {
  int sum = 0;
  if (n == 0)
        return 1;
   for (int i=0; i<n; i++)
       sum++;
  return sum*D(n/2);
}
```

3. [Templates, 52 points]
You are provided with a partial implementation of a templated singly-linked list in *LinkedList.h,* with some missing functionality. It contains a templated *Node* class and a templated *LinkedList* class. **Do not modify the class definitions**.

The linked list class contains the following methods:

- LinkedList – Constructor of a linked list with a head pointing to null (implemented).
- ~LinkedList – Destructor of a linked list.
- deleteFromHead – Removes and returns content of the first node of the list. (If the list is empty, does nothing.)
- deleteFromTail – Removes and returns content of last node of the list. (If the list is empty, does nothing.)
- deleteNode – Removes node with provided data from the list.
- InsertToHead – Inserts node with provided data at the head (implemented).
- InsertToTail – Inserts node with provided data at the tail.
- getSize – Returns the number of nodes in the linked list.
- print - Prints the linked list (implemented).

Notice that while the declaration and implementation of classes are typically split between .cpp and .h files, some compilers cannot handle templates in separate files, which is why we include both the declaration and implementation in a single .h file.

The *main.cpp* file consists of sample test code for each of the tasks below. Your code, which should be added to the provided *LinkedList.h* will be compiled and run with variations of this file. Submit your modified *LinkedList.h* file only.

Your code will be graded based on:

★ How easy it is to read and understand

★ Whether it demonstrate good design and style

a. Complete the implementation of the templated singly-linked list provided in the header file.
b. Implement *StackFromList*, a templated stack class backed by the above singly-linked list. The stack should have a private linked list member, and utilize the linked list methods to implement its functionality. The stack should include a constructor, a destructor, a push, a pop, and an *isEmpty* method (which returns a bool).
c. Implement, *QueueFromList,* a templated queue class backed by the above singly-linked list. The queue should have a private linked list member, and utilize the linked list methods to implement its functionality. The queue should include a constructor, a destructor, an enqueue ~~(insert to head)~~, a deque ~~(remove from tail)~~, and an *isEmpty* method (which returns a bool).