Homework 3

Out: 9.28.23
Due: 10.11.23

1. [Proof by induction, 10 Points]
Prove by induction that the Fibonacci function F(n), defined below, satisfies $F(n) < 2^n$.

$$F(n) = \begin{cases} 1 & if\ n = 1 \\ 1 & if\ n = 2 \\ F(n-1) + F(n-2) & if\ n > 2 \end{cases}$$

2. [Recurrences I, 20 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. $T(n) = 12T(n/11) + n \log n$
b. $T(n) = T(n-1) + 20 \log n$
c. $T(n) = aT(n/a) + n^2 \log n$, for some constant $a>1$
d. $T(n) = 8T(n/2) + 5n^3$

3. [Recurrences II, 20 points]
For each of the following functions, provide:
(1) 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)
(2) 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*n; j++)
      sum ++;
  return sum;
}
```

b.
```
int B(int n) {
  if (n == 0) return 1;
  else return 2*B(n-1);
}
```

c.
```
int C(int n) {
  if (n == n/2)
        return 1;
  else
       return 12*C(n/120);
```

```
}
```

d.
```
int D(int n) {
        int accumulator = 0;
        if (n == 0)
          return 1;
        else {
          for (int i=0; i<n; i+=2)
             for (int j=0; j<n; j+=2)
                accumulator ++;
        }
        return accumulator + D(n/3) + D(n/3) + D(n/3);
}
```

4. [Templates, 50 points]
You are provided with a partial implementation of a templated doubly-linked list in *LinkedList.h*. It contains a templated *Node* class and a templated *LinkedList* class. The list nodes contain a data field, which doubles as key, and pointers to all directions (i.e. to the next node, previous node, node above, and node below). For this assignment, you will only be utilizing the next and previous pointers. Your code will be tested with numerical data values and characters. **Do not modify the class definitions**.

Note: While the declaration and implementation of classes are typically split between *.cpp* and *.h* files, some compilers cannot handle templates in separate files, therefore the declaration and implementation are provided in a single *.h* file.

Implement an ordered doubly linked list (with keys in increasing order). The linked list class should contain the following methods:

- LinkedList – Constructor of a linked list containing two elements: a minimum value and a maximum value (corresponding to -∞ and +∞) of the given type, both should be passed to the function.
- ~LinkedList – Destructor of a linked list.
- search – Accepts 'location', a pointer to a node in the linked list to start a forward search from (such as the head of the linked list, if we are to search the entire list), and 'data' to search for. Tests if 'data' is contained in the linked list, starting from the provided location. If so, it returns a pointer to the node containing 'data', otherwise, it returns a pointer to the node preceding the location where a new node containing 'data' would be inserted.
- insert – Accepts 'location', a pointer to a node in the linked list, and 'data' to be inserted. Creates and inserts a node containing 'data' directly after the node in the specified 'location', and returns pointer to the new node. Note that search will generally be called first, return the specified location for inserting the new node, and then insert will be called. You may assume that no duplicates are allowed. We assume that the provided location is the correct position to insert 'data', but if this

is not the case (i.e. the resulting list is not sorted), then the new node should not be inserted, and the function should return NULL.

- printData – Prints the content of the linked list (data values only, separated by spaces), utilizing the node class' printData method.
- print – Print the entire content of the linked list (including nodes' addresses and pointers), utilizing the node class' print method, may be used for debugging purposes.

The *printData* function in the provided *main.cpp* should give the following output:

Linked list data:
-2147483648 1 2 3 2147483647

Submit your modified *LinkedList.h* file only. Your code must compile and run on the lab computers with a variation on the provided *main.cpp* file.
Your code will be graded based on:

★ Correct functionality
★ How easy it is to read and understand
★ Whether it demonstrate good design and style