CS 3358 Section 252 – Assignment 3

Due Date: March 31, 2023

This assignment has four parts pertaining to the recursion implementation. The parts of code are given in the `.cpp` and `.h` files. The places you need to fill out in the code are marked by `// TODO`.

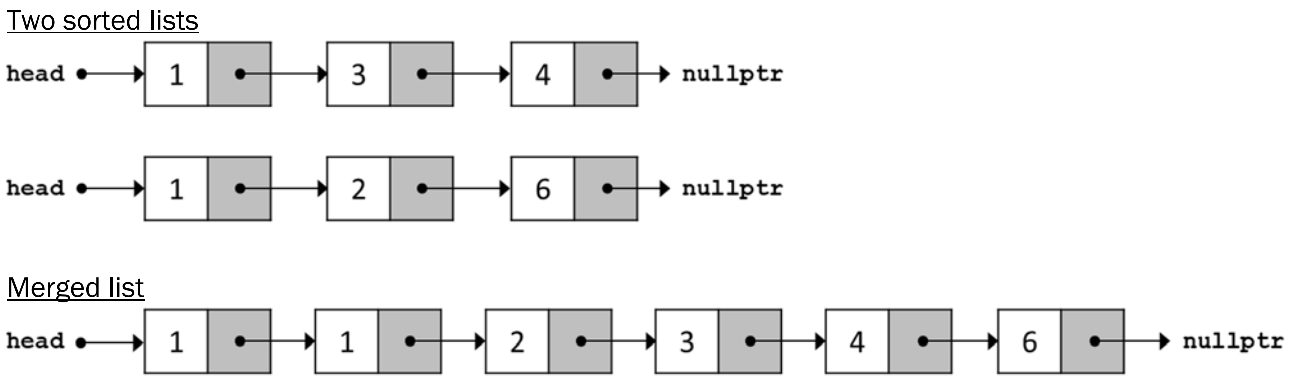
In this assignment, you are asked to implement four recursive functions. Note that you are not allowed to use iterative implementations, and you will not receive any credit for the iterative implementations. You are generally expected to implement the functions whose headers are given in the code files, and you are not expected to implement additional helper functions to implement them.

- (15 points) In `mypow1.cpp`, implement the recursive function `Pow` to compute the value of a power function x^n based on the basic recursion method that you learned from class, where the inputs are a number `x` and an integer $n \geq 0$, and the output is the value of x^n. Note that the basic recursion method uses the definition, i.e., $x^n = x*x^{n-1}$ for $n \geq 1$ and $x^n = 1$ for $n = 0$ and leads to the O(n) run-time.

- (15 points) In `mypow2.cpp`, implement the recursive function `ImprovedPow` to compute the value of x^n based on the improved recursion method using repeated squaring that you learned from class, where the inputs are a number `x` and an integer $n \geq 0$, and the output is the value of x^n. Note that the improved recursion method leads to the O(log n) run-time.

- (30 points) In `poweroftwo.cpp`, implement the recursive function `CheckPowerOfTwo` to check if a given non-negative integer `n` is a power of 2. Return `true` if it is a power of 2. Otherwise, return `false`.

- (40 points) In `listnode.h`, implement two member functions of the class `ListNode` such as the destructor and the function `AddNode` (i.e., appending a node to the end of the list), and in `mergetest.cpp`, complete the implementation of the function `MergeLists` for the following problem: Give two sorted linked lists, merge them as a single sorted list. It should be done by joining together the nodes of the lists, and no sorting algorithm is required. The following is an example.



Submission:
You should submit your work via Canvas. You should pack `mypow1.cpp`, `mypow2.cpp`, `poweroftwo.cpp`, `mergetest.cpp`, and `listnode.h` into a single `.zip` file to upload to Canvas. The `.zip` file should be named as `a3_yourNetID.zip`, such as `a3_gwc38.zip`.

Test cases for `poweroftwo`:

```
Enter an integer (0,1,...): 8
Power of two: true
Enter an integer (0,1,...): 64
Power of two: true
Enter an integer (0,1,...): 1
Power of two: true
Enter an integer (0,1,...): 5
Power of two: false
Enter an integer (0,1,...): 0
Power of two: false
```

Test cases for `mergetest`:

Test case 1
```
Type in an integer to insert to list1, e.g. 1,2,3,... (-1 to quit): 10
13
16
-1
Type in an integer to insert to list2, e.g. 1,2,3,... (-1 to quit): 2
20
27
-1
List 1: 10 13 16
List 2: 2 20 27
Merge List: 2 10 13 16 20 27
```

Test case 2
```
Type in an integer to insert to list1, e.g., 1,2,3,... (-1 to quit): 1
3
4
-1
Type in an integer to insert to list2, e.g., 1,2,3,... (-1 to quit): 1
2
6
-1
List 1: 1 3 4
List 2: 1 2 6
Merge List: 1 1 2 3 4 6
```

Test case 3
```
Type in an integer to insert to list1, e.g., 1,2,3,... (-1 to quit): -1
Type in an integer to insert to list2, e.g., 1,2,3,... (-1 to quit): -1
List 1: List is empty
List 2: List is empty
Merge List:
```