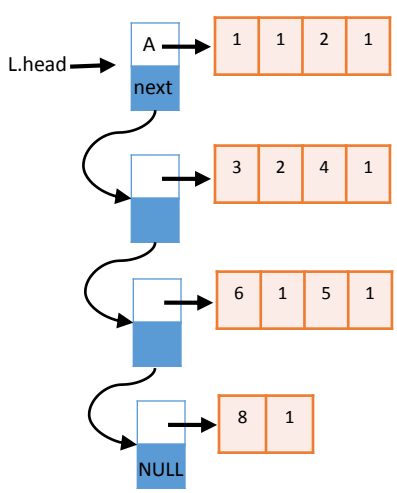
Fall 2018, CMPSC 465
Homework Assignment #4

This homework is due **9 pm** on **October 9th**.
Collaboration is permitted on this homework. If you choose to collaborate, you are allowed to discuss each problem with at most three other students currently enrolled in class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. *You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem.* You must also identify your collaborators. If you do not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or a TA. *Finding answers to problems on the web or from other outside sources (includes anyone not enrolled in the class) is strictly forbidden.*
You should aim to be as clear and concise as possible in your writeup of solutions. A simple and direct analysis is worth more points than a convoluted one. Each problem is worth 10 points. Points may be deducted for illegible handwriting. Partial credit will be given only for answers that make significant progress toward correct solution.

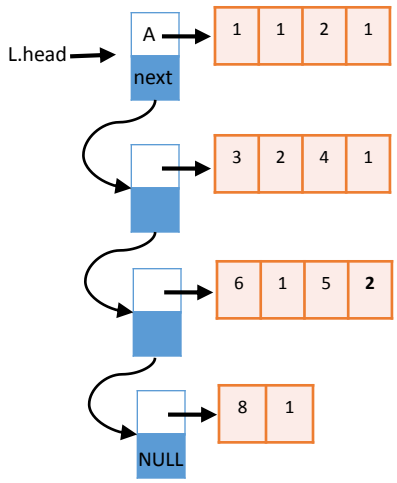
1

Consider the following *hybrid* linked list-array data representation for storing integers in $[1, C]$. We use a linked list of size B objects or nodes to store the integers. Each list node points to an (initially-empty) dynamic array. The ith list node holds keys lying in an interval of width C/B. The dynamic array is resized by doubling the size with new insertions, i.e., the array size would grow as $2, 4, 8, 16, \ldots$. The arrays hold integer keys (and their counts in the adjacent cell) in unsorted order.

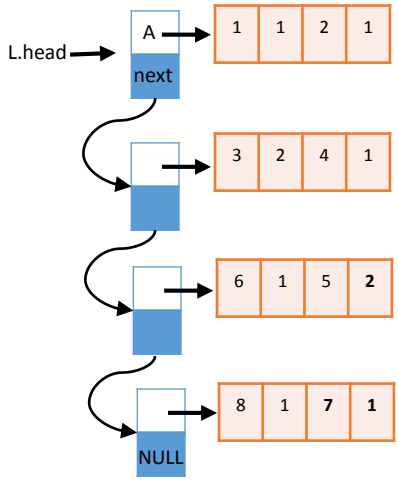
For example, consider $n = 8$, $B = 4$, $C = 8$, and the numbers $1, 3, 3, 2, 4, 6, 5, 8$ currently stored in the data structure. Graphically,the data structure would look like the following:



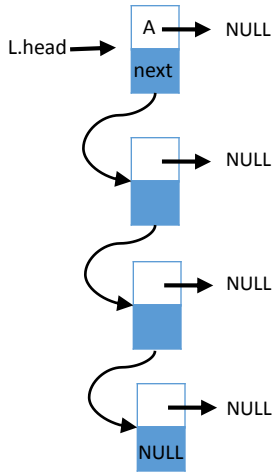
Now, consider inserting key 5. The updated structure would be



Now, consider inserting key 7. The updated structure would be



Note that the data structure initially looks like the following, when $n = 0$:



- Write pseudocode for testing membership of key k. If the key is present, the algorithm must return the count. If not, it must return 0. What are the best-case and worst-case times for membership queries (in terms of n, C, B), given that there are n elements already stored in the structure?
- Write pseudocode for inserting a key k that is known to be already present in the structure. What are the best-case and worst-case times for this operation, given that there are n elements already stored in the structure?
- Write pseudocode for inserting a new key k into the structure. What are the best-case and worst-case times for this operation, given that there are n elements already stored in the structure?

2

- Consider a sequence of +1's and -1's with the property that the sum of any prefix of the sequence is never negative. For example, the sequence $+1, -1, +1, -1$ satisfies this property, but $+1, -1, -1, +1$ does not, since the prefix $+1 - 1 - 1 < 0$. Describe any relationship between such a sequence and a Stack push(x) and pop() operations.
- A *matched string* is a sequence of {, }, (,), [, and] characters that are properly matched. For example, "{{()[]}}" is a matched string, but this "{{()]}" is not, since the second { is matched with a]. Show how to use a stack so that, given a string of length n, you can determine if it is a matched string in $O(n)$ time.

3

Describe a simple array-based implementation of the List ADT. What are the asymptotic running time costs of various List ADT operations? For simplicity, ignore array resizing costs.

4

- Illustrate the operation of MAX-HEAP-INSERT(A, 9) on the the heap $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 3, 6, 2, 1 \rangle$.
- Give pseudocode for the routine MIN-HEAP-DELETE(A, k). Assume that key k is at location l of the heap and that $1 \leq l \leq n$.
- Show how heapsort would work for the input array $A = \langle 5, 7, 3, 11, 1, 2 \rangle$.

5

- Describe how to add the elements $\{1, \ldots, n\}$ to an initially empty binary search tree in such a way that the resulting tree has height $n - 1$. How many ways are there to do this?
- Prove that, if a binary tree, T, has at least one leaf, then either (a) T's root has at most one child or (b) T has more than one leaf.

6

Consider inserting the keys 10, 12, 34, 4, 15, 28, 17, 88, 59 into a hash table of length $N = 11$ using open addressing with the hash function $h(k) = k \bmod N$. Illustrate the result of inserting the keys using linear probing.

7

- Consider a BST T whose keys are distinct. Show that if the right subtree of a node x in T is empty and x has a successor y, then y is the lowest ancestor of x whose left child is also an ancestor of x.
- Give pseudocode for the TREE-PREDECESSOR procedure.