Kamryn Parker

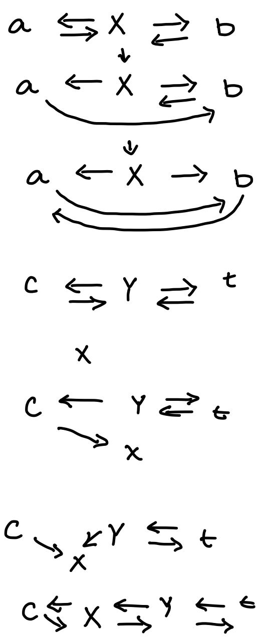
CS 321 Data Structures (Spring 2020)

Homework #2 (86 points), Due date 4/010/2020 (Friday)

- **Q1(24 points): Basic Data Structures: Stacks, Queues, Linked Lists and Trees**

(a)(8 points) Please write a pseudocode for `List-move(x, y)`. This procedure is to move an existing node x to the front of another existing node y in a doubly non-circular list without a dummy head.

a or b can be null potentially

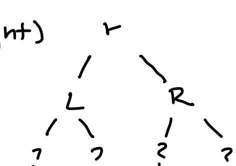



```
if x.previous is not null
    X.previous.next = x.next
if x.next is not null
    x.next.previous = x.previous
 if Y.previous is not null
    Y.previous.next = X
 X.previous = Y.previous
 Y.previous = X
 X.next = Y
```

(b)(8 points) Given a binary tree T with a root node r, please write a recursive method `Nodes1Child(r)` that returns the number of nodes with only one child in the tree.

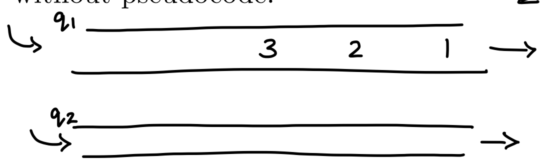
```
if r == null
    return 0
if r.left does not exist and r.right does not exist
    return 0
if r.left does not exist or r.right does not exist
    return Nodes1Child(r.left) + 1 + Nodes1Child(r.right)

return Nodes1Child(r.left) + Nodes1Child(r.right)
```



(c)(8 points) How to use two queues to implement a stack so that push runs in $O(1)$ and pop runs in $O(n)$? Suppose the queues have no size limit. Please describe your algorithm without pseudocode.

Normally FIFO



can't pull from middle

All push can do is enqueue it into a queue
q1 is data storage so push will be enqueueing the element to q1

The top of a stack is the back of a queue essentially
You would dequeue elements from q1 and enqueue them into q2 until there is one element left in q1. Then you would dequeue the last element and hold it in a temp variable for now. Then you would swap q1 and q2. Then finally you would return that temp variable which is returning the element we needed to remove.

This is a linear O(n) because you have to dequeue n-1 times to get to the final element then poping that element

- **Q2(20 points): Hashing**
Suppose we would like to insert a sequence of numbers into a hash table with table size 8 using the three open addressing methods, with the primary hash function $h_1(k) = k \bmod 8$, the secondary hash function $h_2(k) = 1 + (k \bmod 7)$, and the constants $c_1 = c_2 = 1/2$ (in quadratic probing).

(a)(10 points) If the sequence of numbers is $< 63, 41, 23, 15, 31 >$, please successively insert these numbers into the following tables.

23 mod 8 = 7 + 0 + 0
= 7 + 1/2(1) + 1/2(1)^2
= 8 mod 8 = 0
15 mod 8 = 7
= 7 + 1/2(2) + 1/2(2)^2 = 10
10 mod 8 = 2
31 mod 8
7 + 1/2(3) + $\frac{1}{2}$(3)^2 = 13
13 mod 8 = 5

index	linear		quadratic		double
0	23		23		
1	41		41		41
2	15		15		15
3	31				23
4					31
5			31		
6					
7	63		63		63

23 mod 8 = 7 − 1 = 6
1 + (15 mod 7)

(b)(3 points) For the hashing functions and table size we used in part (a), does the linear probing fully utilize the table? How about the quadratic probing and double hashing ?

No linear probing for this example has created primary clustering where there are a long run of occupied slots. The quadratic still has a good amount of collision errors so it is not as good either. Using the double hash functions cuts down on multiple collisions so I would say it does better to utilize the table size

(c)(7 points) A hash table with size 10 stores 6 elements. These 6 elements are stored in T[0], T[1], T[3], T[6], T[7], T[9]. Suppose that all the other entries contain no "deleted" flag. An entry has a "deleted" flag means that this entry stored an element before, but the element has already been deleted. If we would like to search an element with a key k and assume the linear probing technique is used, what is the expected number of probes for an unsuccessful search?

load = $\frac{n}{m}$ $\quad \frac{6}{10}$

$\frac{1}{(1-\alpha)}$ $\quad \frac{1}{(1-(\frac{6}{10}))}$ = 2.5 expected probes

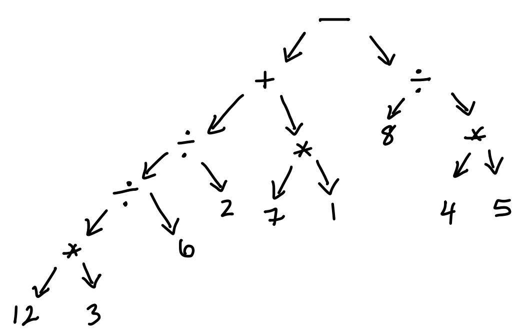
we could round up to 3 for $\lceil 2.5 \rceil$ procedure if needed

- **Q3(12 points): Expressions and Expression Trees**

  For a given in-fix expression as below:

$$12 * 3 \div 6 \div 2 + 7 * 1 - 8 \div 4 * 5$$

(a)(6 points) What is the corresponding expression tree?



$((((12 * 3) \div 6) \div 2) + (7 * 1) - ((8 \div 4) * 5)$

(b)(6 points) What are the corresponding pre-fix and post-fix expressions?

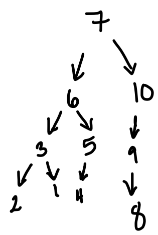
Pre-fix = $- + \div \div * \ 12 \ 3 \ 6 \ 2 \ * \ 7 \ 1 \ \div \ 8 \ * \ 4 \ 5$

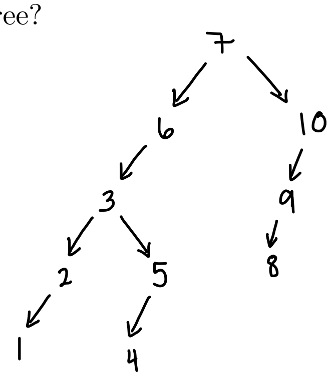
Post-fix = $12 \ 3 \ * \ 6 \ \div \ 2 \ \div \ 7 \ 1 \ * \ + \ 8 \ 4 \ 5 \ * \ \div \ -$

- **Q4(10 points): Binary Search Trees**

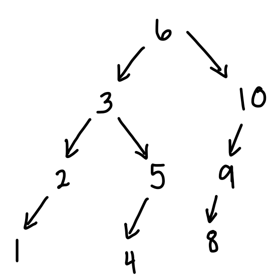
  For a given input array $A : < 7, 10, 6, 9, 3, 5, 2, 4, 1, 8 >$,

  (a)(6 points) What is the resulting binary search tree after inserting the numbers in the list to an initially empty tree?





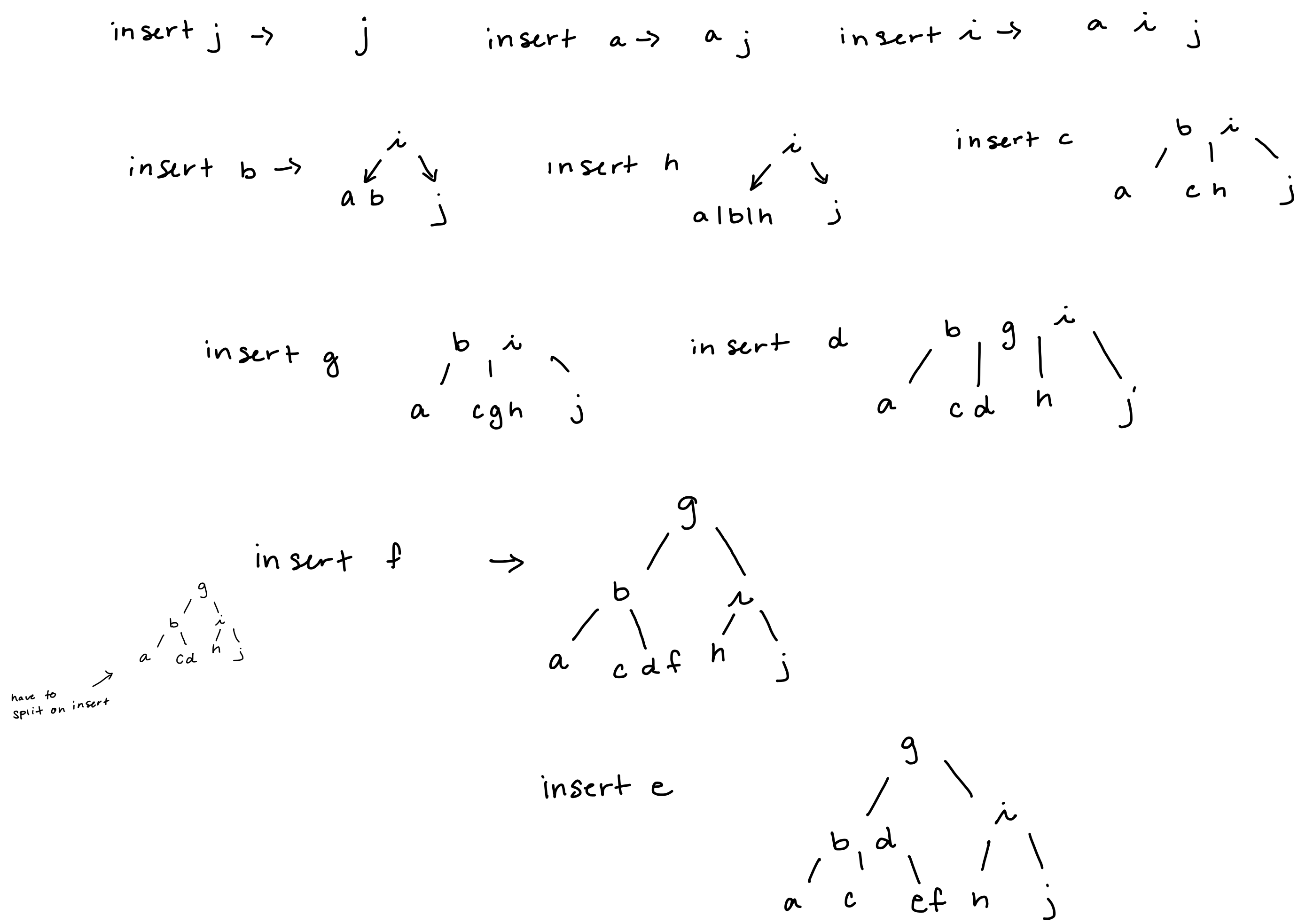
  (b)(4 points) From the tree you have built in part (a), what is the resulting tree after deleting the value 7 ?



6 would move to the top as the root

- **Q5(20 points): B Trees**

  (a)(10 points) For a sequence of keys {`j, a, i, b, h, c, g, d, f, e`}, suppose we would like to construct a `B-Tree`, with degree 2, by successively inserting those keys, one at a time, into an initially empty tree. Please draw the sequence of `B-Trees` after inserting each of the 10 keys.
  Note: please draw only one tree after each insertion.



Traversal check

a b c d e f g h i j

(b)(5 points) Let t be the (minimal) degree of a BTree. Suppose the size of each object, including the key, stored in the tree is 60 bytes. Also, suppose the size of a BTreeNode pointer is 4 bytes. In addition, 40 bytes of meta-data is required for each BTree node to keep track of some useful information. Suppose each BTreeNode has only the meta-data, a parent pointer, a list of objects, and a list of child pointers. What is the optimal (minimal) degree for this BTree if a disk block is 4096 bytes?

largest we can make the degree so largest node is less than disk block

$2t$ = children

$2t-1$ = num objects

meta data → 40

parent pointer → 4 or $(2t+1)(4)$ ← if child and parent combined

list of objects → $(2t-1)(60)$

list of child pointers → $(2t)(4)$

$$4096 \geq 40 + 4 + (2t-1)(60) + (2t)(4)$$

$$4096 \geq 40 + 4 + 120t - 60 + 8t$$

$$4096 \geq -16 + 128t$$

$$4112 \geq 128t$$

$$t \leq \lfloor 32.125 \rfloor$$

$t = 32$ ← highest integer that meets the inequality

(c)(5 points) For a BTree with height 4 (or 5 levels), what is the maximal number of objects can be stored if the (minimal) degree $t = 80$?

$$\sum_{i=0}^{4} (160)^i = 659\,481\,761$$

$m = 2t$

$m = 2(80)$

$m = 160$