Trevor Smith

CS 321 Data Structures (Fall 2020)

Exam #2 (100 points), 11/18/2020 (Wednesday)

- **Q1(27 points): Stacks, Queues, Linked List and Trees**

(a)(10 points) Please write a pseudocode for `List-concate(`h_1, h_2`)`. This procedure is to concatenate a linked-list L_2 to the end of another linked list L_1. Assuming both lists are "non-empty" and are doubly circular lists without dummy heads. h_1 and h_2 are the head pointers to L_1 and L_2 respectively. After the procedure, the concatenated list is again a doubly circular list without a dummy head.

```
List-concate(h1, h2)
{
```

1. Node t1 = h1.prev;
2. Node t2 = h2.prev;
3. h1.prev = t2;
4. t2.next = h1;
5. t1.next = h2;
6. h2.prev = t1;

```
}
```

(b)(7 points) Given a binary tree rooted at a node r, please write a recursive pseudocode to count the number of internal nodes with only one child in the tree.

```
NodesW1C(r)   // r is the root of a binary tree
              // return the number of internal nodes with exactly one child
{
  if ((r == null) || (r.left == null && r.right == null))

     return ______________________0_____________________________;

  if (r.left != null && r.right != null)

     return ______________________0_____________________________;

  return 1+NodesW1C(r.left)+NodesW1C(r.right);
}
```

(c)(10 points) How to use two queues q_1 and q_2 to implement a stack so that push runs in $O(n)$ and pop runs in $O(1)$? Suppose both stacks have no size limit. Please describe your algorithm without pseudocode.

In order to implement a two queue stack with push O(n) and pop O(1), the algorithm would be as follows:
Given queues q1 and q2,

Push: Enqueue new element into q2, then dequeue each element from q1 and enqueue to q2 until q1 is empty. After this, you would then switch the names of q1 and q2.

Pop: simply dequeue from q1 and return.

- **Q2(20 points): Hashing**
  Suppose we would like to insert a sequence of numbers into a hash table with table size 7 using the three open addressing methods, with the primary hash function $h_1(k) = k \bmod 7$, the secondary hash function $h_2(k) = 1 + (k \bmod 6)$, and the constants $c_1 = c_2 = 1/2$ (in quadratic probing).

  (a)(10 points) If the sequence of numbers is $< 47, 54, 19, 12, 26 >$, please successively insert these numbers into the following tables.

index	linear	quadratic	double
0	19		19
1	12	19	12
2	26		
3			
4		12	26
5	47	47	47
6	54	54	54

  (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 ?

  Linear probing fully utilizes the table. Quadratic probing does not since m is not a power of 2. Double hashing fully utilizes the table since each h2(k) value is relatively prime to 7.

  (c)(7 points) A hash table with size 10 stores 6 elements. These 6 elements are stored in T[0], T[2], T[3], T[4], T[8], T[9]. Suppose that all 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?

0	A
1	
2	B
3	C
4	D
5	
6	
7	
8	E
9	F

$$\frac{4}{10}\times 1 + \frac{2}{10}\times 2 + \frac{2}{10}\times 3 + \frac{2}{10}\times 4$$

$$\frac{4}{10} + \frac{4}{10} + \frac{6}{10} + \frac{8}{10} = \frac{22}{10} = 2.2$$

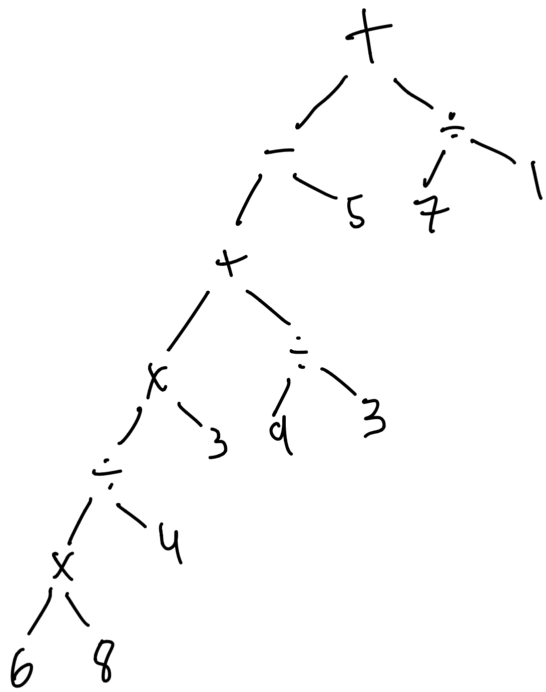
$\boxed{\text{2.2 probes}}$

- **Q3(20 points): Arithmetic Expressions and Expression Trees**

  For a given infix arithmetic expression as below:

$$6 \, * \, 8 \, \div \, 4 \, * \, 3 \, + \, 9 \, \div \, 3 \, - \, 5 \, + \, 7 \, \div \, 1$$

(a)(10 points) What is the corresponding arithmetic expression tree?



(b)(10 points) What are the corresponding pre-fix and post-fix arithmetic expressions?

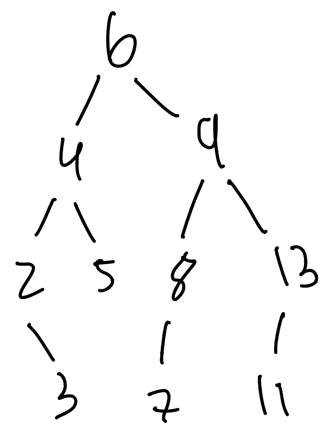
pre-fix : + - + x ÷ x 6 8 4 3 ÷ 9 3 5 ÷ 7 1

post-fix : 6 8 x 4 ÷ 3 x 9 3 ÷ + 5 - 7 1 ÷ +

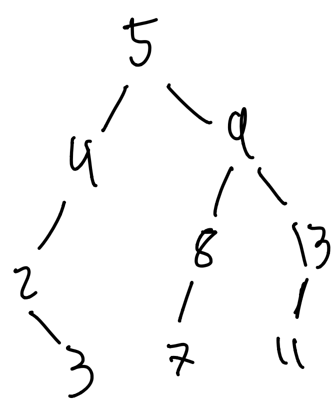
- **Q4(10 points): Binary Search Trees**

  For a given input array $A : < 6,\ 9,\ 8,\ 4,\ 13,\ 5,\ 2,\ 7,\ 3,\ 11 >$,

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

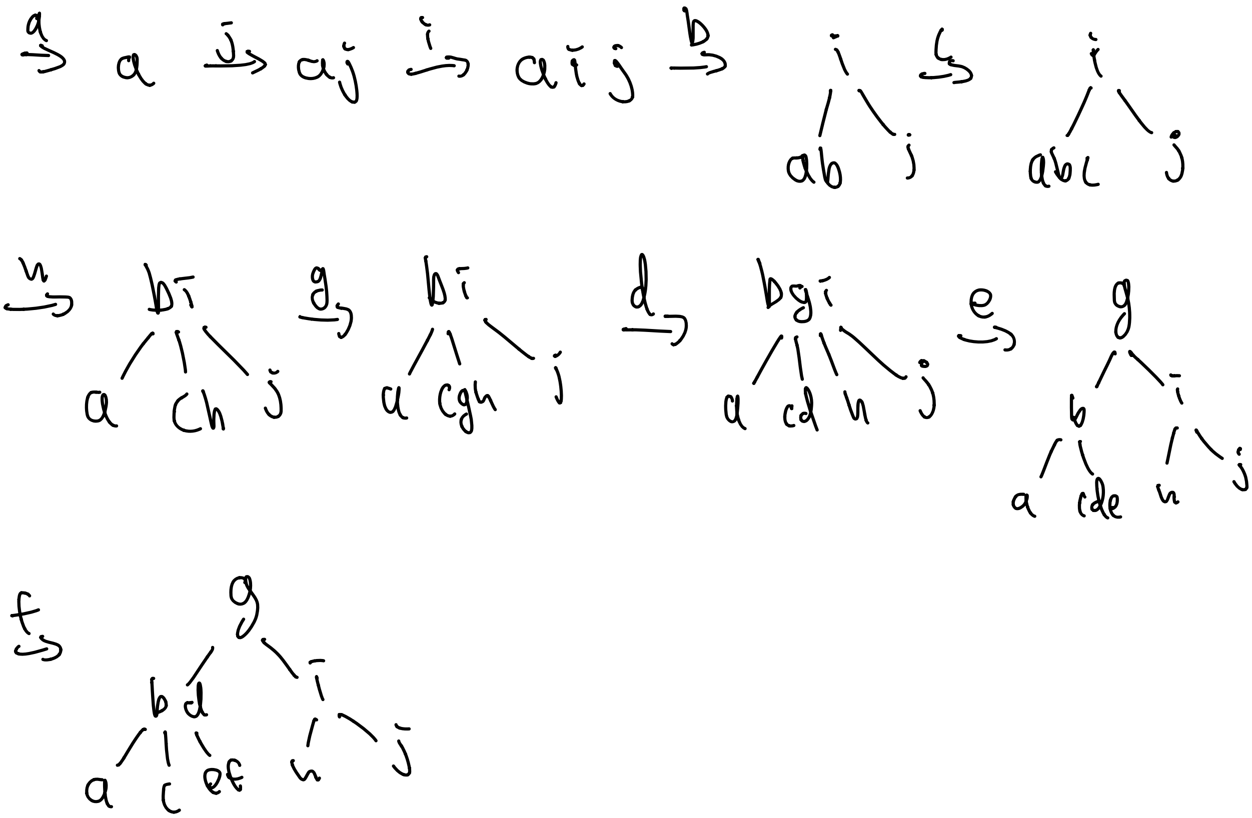


  (b)(4 points) From the tree you have built in part (a), what is the resulting tree after deleting the value 6 ?



- **Q5(23 points): B Trees**

  (a)(10 points) For a sequence of keys {`a, j, i, b, c, h, g, d, e, f`}, 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.



(b)(6 points) Suppose the size of each object, including the key, stored in a B-tree is 35 bytes. Also, suppose the size of a BTreeNode pointer is 4 bytes. In addition, 50 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 degree t for this BTree if a disk block is 4096 bytes?

$$50 + 35(2t-1) + 4(2t+1) \le 4096$$

$$70t - 35 + 8t + 4 \le 4046$$

$$78t \le 4077$$

$$\boxed{52 = t}$$

(c)(7 points) For a BTree with height 4 (or 5 levels), please find the (minimum) degree t such that the tree can always store more than 100,000,000 objects.

$$(2t)^5 - 1 = 100{,}000{,}000$$

$$t = 19.905$$

$$\text{ceil}(19.905)$$

$$\boxed{t = 20}$$