Computer Science 260 Practice for the Midterm3
Out of XX possible marks NAME:_____

1. (8 marks) Circle either **T** (True) or **F** (False), whichever is a more accurate assessment of the following statements:

T **F** When open hashing, the number of slots must be at least as great as the number of keys to be stored.

T **F** To rebalance an AVL tree after an insertion, we look for the highest node that is out of balance and do rotations at that node first.

T **F** A good hash function guarantees that the keys will distribute uniformly among the slots.

T **F** All the Minimum Spanning Tree (MST) algorithms we studied are examples of greedy algorithms.

2. (6 marks) Use Dynamic Programming to find the longest common subsequence (not necessarily contiguous) in the following two strings, by filling the table below.

	λ	B	B	A	B	C	A	B	D
λ	0	0	0	0	0	0	0	0	0
A	0	0	0	1	1	1	1	1	1
B	0	1	1	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3	3
C			1	2	2	3	3	3	3
C	0	1	1	2	2	3	3	3	3
A	0	1	1	2	2	3	4	4	4
D	0	1	1	2	2	3	4	4	5
B	0	1	2	2	3	3	4	5	5

BACAD

Advice: if Q asks to "use Dynamic Programming" to "find the LCS" then the grader needs to see two things:

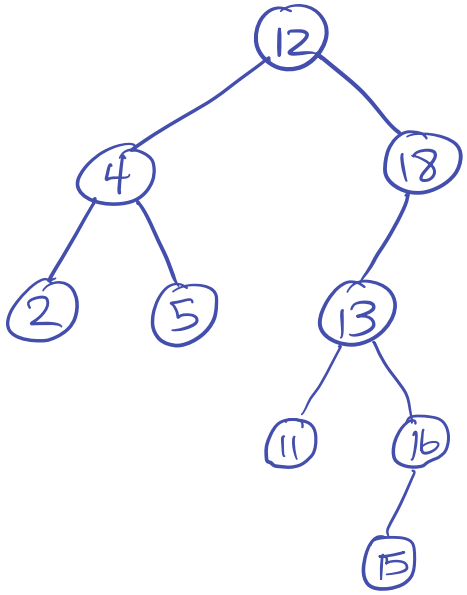
1. The LCS (Longest Common Subsequence)
2. The table, filled correctly, to demonstrate that DP has been used.

Recall, for LCS, DP fills each table entry with

$LCS[i,j] = \max(LCS[i-1,j], LCS[i,j-1]$, and

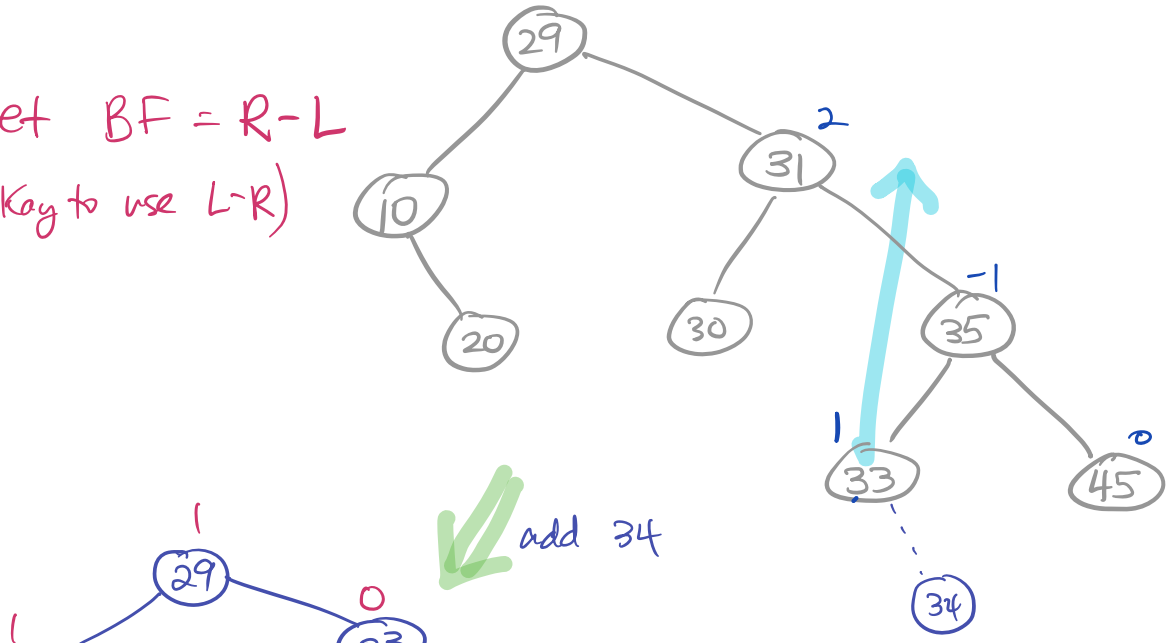
possibly 1 + LCS[i-1, j-1], if X[i] = Y[j] ie letters match)

3. (5 marks) Perform the BST inserts of 12, 18, 13, 4, 5, 2, 11, 16, 15 into an originally empty BST. Do them in that order; show the tree that results. You only need to show the end result.

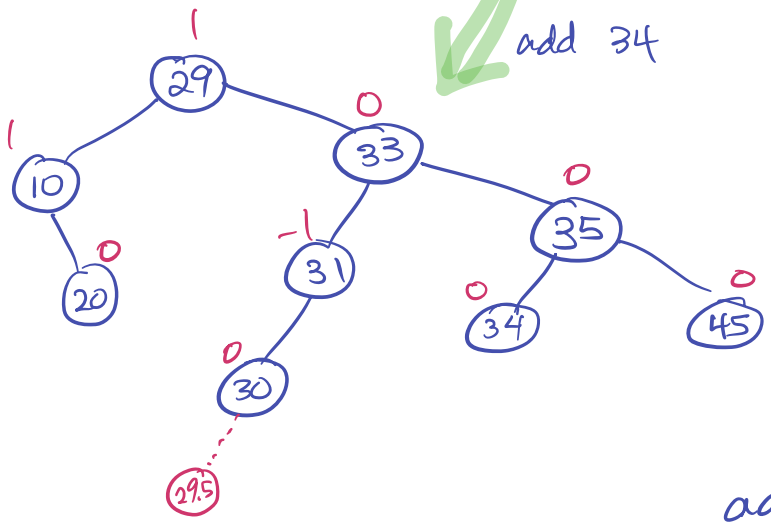


4. (5 marks) Execute an AVL-Tree insert of key 34 into the following AVL tree. Then label each node with its BF value. Then AVL-insert 32.

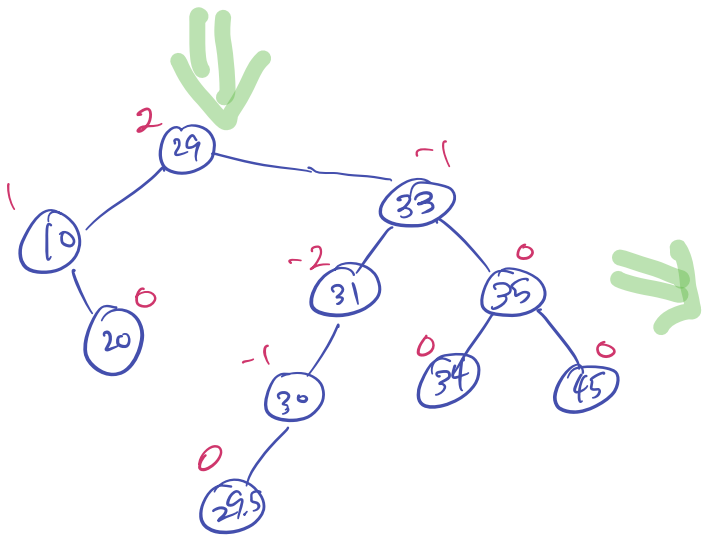
Let BF = R-L
(okay to use L-R)



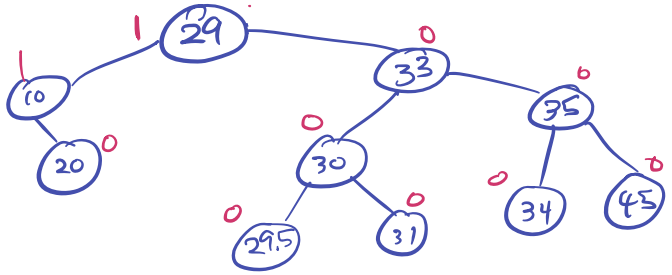
add 34



add 32
- not very interesting.
Let's make the question a little harder ...
add 29.5



Lowest node that is out of Balance is (31)



5. (4 marks) Define 'clustering' and describe a strategy for avoiding it when using closed hashing.

Clustering: when keys concentrate in an area of the table, and collisions increase as the cluster grows.

To avoid: use a secondary hash function.

6. Consider the hash function $h(c_1c_2\ldots c_n) = \Sigma_{i=1}^{n}\text{ord}(c_i)\text{mod}m$. Note that $\text{ord}(c_i)$ is the ordinal number of the letter c_i in the alphabet. The ordinal values are provided here for your convenience:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

(a) (1 mark) What is does 'm' refer to in the above hash function? Number of slots

(b) (6 marks) Insert the following keys into the hash table below, using open addressing (all elements are stored in the table itself) with double hashing to handle collisions. The secondary hash function is "string length + number of vowels" mod m. AN MB AR WX DE LN GI UF

0	1	2	3	4	5	6	7	8	9	10
UF			WX	AN	GI	MB	..	AR	DE	LN

h(WX) = 3
h(DE) = 9
h(LN) = 4 collision!
2nd hash is 2. keep adding 2 to 4 until find an empty slot

(c) (2 marks) Why is it a bad idea if the value of m were 12? What kind of behaviour might it lead to in insertion and in deletion?

If m = 12, then it is possible that an insert function would not find an empty slot even if half the slots were empty... if the secondary hash is 2 for example

(d) (3 marks) Write the Find(key) algorithm for the above table. What happens if some elements have been deleted? → Deleted entries leave DELETED in slot, not EMPTY,

```
int Find(int K)  // return slot number containing
                 // key K
  S = h1(K); offset = h2(K); try=0;
  while T[S] != K and try < m and T[S] != EMPTY
      S += offset; try++
  if T[S] = K, return K
  else return NOTFOUND.
```

So algorithm will continue searching as if that slot were filled.

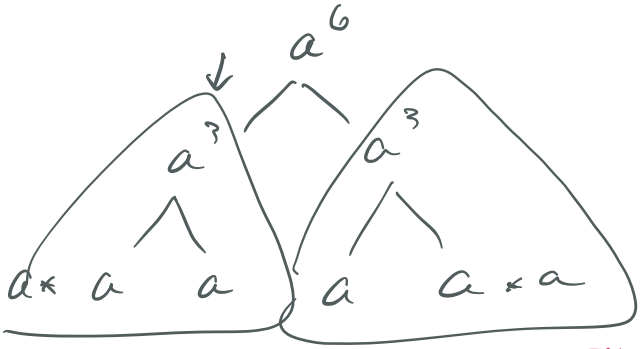
7. (6 marks) Give a simple, efficient, recursive, Dynamic Programming algorithm to perform exponentiation. Hint: $a^n = a * a^{\frac{n}{2}} * a^{\frac{n}{2}}$ if n is odd, and $a^n = a^{\frac{n}{2}} * a^{\frac{n}{2}}$, if n is even. Integer division, rounding down, is used in the exponents.

```
global variable int aToTheN[n+1] is declared and initialized to zeroes.

Algorithm Raise( a, n )
input: a is a floating point number, n is an non-negative integer
output: a^n
```

a^n

$a^{n/2} \cdot a^{n/2}$



$a^7 = a \cdot a^3 \cdot a^3$

Answer not provided ... but Instructor highly recommends students think about how to solve it...

8. Give a set of coin denominations for which the greedy algorithm does not provide the optimal (minimum) number of coins. Prove your claim by giving an amount C, and the change made by the greedy algorithm, and a way to make the change that uses fewer coins.

Denominations = [1, 5, 7]

Change Amount = 10.

Greedy yields 7+1+1+1 or 4 coins; optimal is 2 coins 5+5.

9. Describe how you would efficiently find the record with minimum key value for each of the following Data Structures, and give the asymptotic running time for the algorithm. Your description can be a single line (or more, if necessary) in English, or pseudocode. CLARITY is important! Assume that the insert algorithms are the standard ones, and cannot be changed.

(a) (4 marks) AVL tree of n nodes.

Start at root;
Go left while ∃ a left child.
Return the value at this leftmost node.

Runs in $\Theta(\log n)$ since height of an AVL tree is $\Theta(\log n)$.

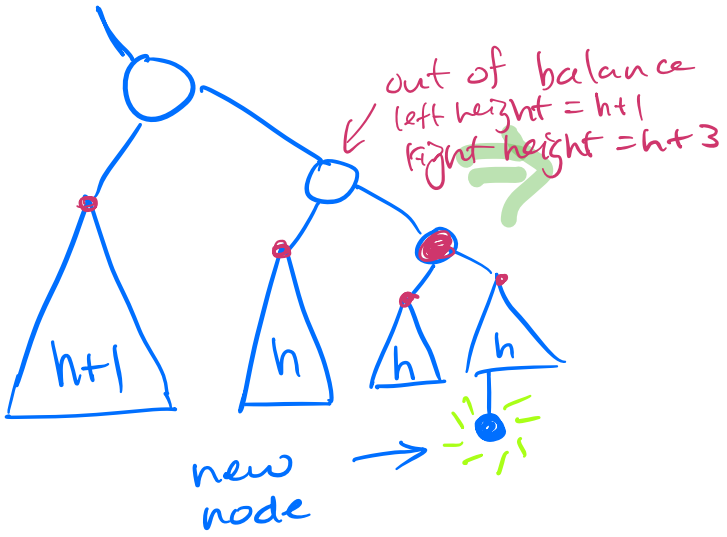
(b) (4 marks) MinHeap of n nodes.

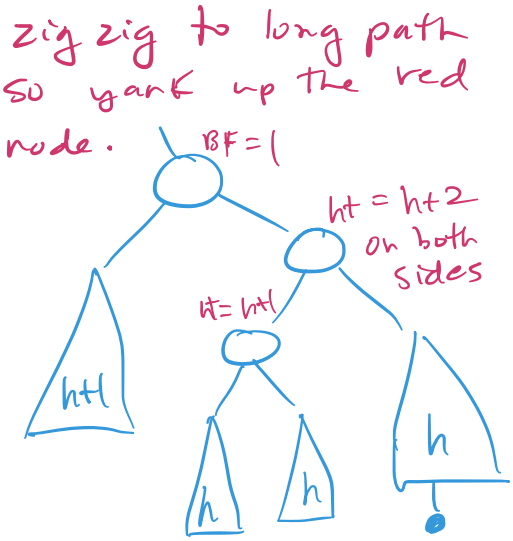
- return pointer to root node.
- $\Theta(1)$.

(c) (4 marks) Hash table, table size m, number of keys in it is n where $n \leq m$, and the Universe of key values is 2^{10}, which is much bigger than m; and Open Addressing (all elements stored in table) is used.

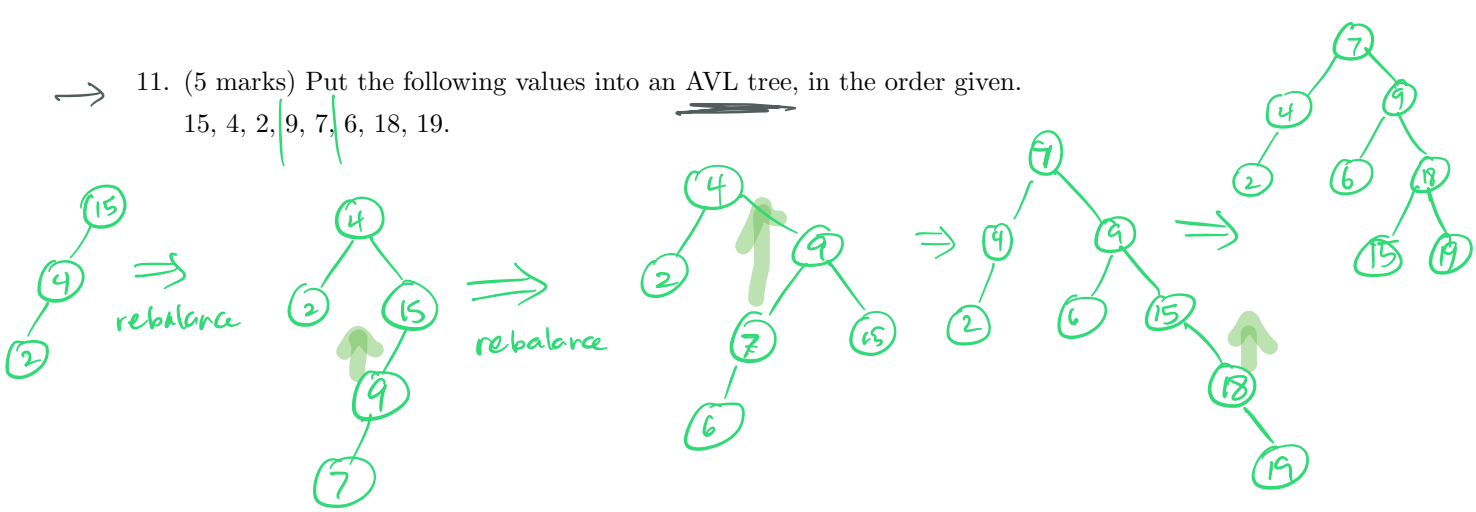
- linear search of hashtable of size m
- $\Theta(m)$

10. (5 marks) Suppose an element is inserted into an AVL tree and its position, before rebalancing, is as shown below, with relative heights as shown. Perform the necessary rebalancing and give the AVL tree, with nodes and subtrees labelled, that results.





11. (5 marks) Put the following values into an AVL tree, in the order given.
15, 4, 2, 9, 7, 6, 18, 19.



12. Hash Tables

(a) (2 marks) What best describes the ADT that hashing is designed to implement: Dictionary, Priority Queue, Graph, Heap, Skip List, Sparse Table, Ink Blot, Binary Counter.

(b) (4 marks) Describe in general terms the strategy for handling collisions utilized by Open Hashing (Separate Chaining).

(c) (4 marks) Describe Open Hashing (Separate chaining). Under what circumstances is it preferred to Open Addressing (where all elements are stored in the table)?

(d) (4 marks) Write the pseudocode for ChainedHashDelete($T, x,$), where T is a hash table, x is the key value of the item to be deleted, and collisions are handled by chaining.

Gara - provide

13. (5 marks) Under the assumption of Simple Uniform Hashing, if x and y are two keys selected from a set of keys K, where K is a subset of the Universe of keys U, and the hash table is of size m, determine the following, in terms of m, $|K|$ and $|U|$.

(a) the probability that x and y hash to the same slot

$\frac{1}{m}$

(g) In the following situation, if Open Addressing with and quadratic probing is used, where $h(k,i) = (h'(k) - 2i + i^2) \text{mod} 11$, show the result of inserting A, B, C, and D, in that order, if $h'(A) = 4, h'(B) = 6, h'(C) = 10, h'(D) = 3$.

No collisions? Trivial. Try it if they all hash to same slot.

```
ChainedHashDelete(T, x) // assume singly linked chains.
    s = h(x)
    p = T[s]   // p points to start node of
               // chain containing x, if exists
    prev = p
    while (p != NULL and p->key != x)
        prev = p
        p = p->next

    if p == NULL return NOT FOUND
    if p = prev // front of chain
        T[s] = p->next
        return p
// x is found, and not at start of chain:
```

```
prev -> next = p -> next;
return p
```