ECE 309 Exam 2 - Fall 2019 A

Name (print): ________________________ UnityId (print) ________________@ncsu.edu

Exam 2 Solution

This is a closed book and closed notes test. You are not allowed to have anything on your desk other than pencil and this exam paper during the test; this includes *calculators* or *electronic assistance* of any kind – ***especially smartphones***.

You may not leave to go to the restroom. Please go before the exam starts.

You may not ask questions. If something is confusing, write a note beside the question and explain your assumptions.

You must show all of your work on this exam. You will not be allowed to turn in additional sheets of paper.

Read and sign the following statement. Failure to sign the statement will result in a **zero** on the exam.

I have neither given nor received unauthorized assistance on this test. I have notified the proctor of any violations of the above policies.

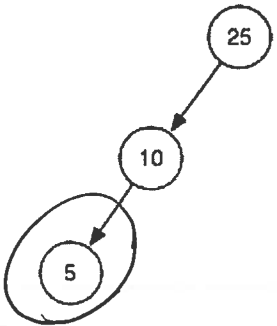
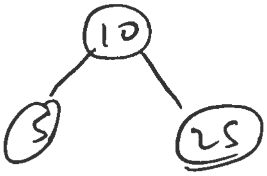
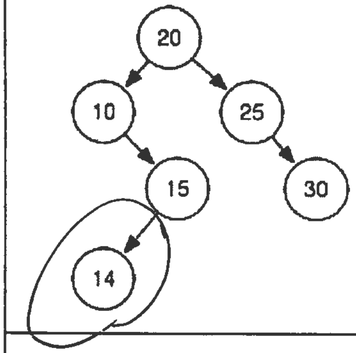
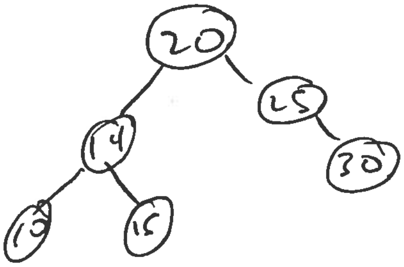
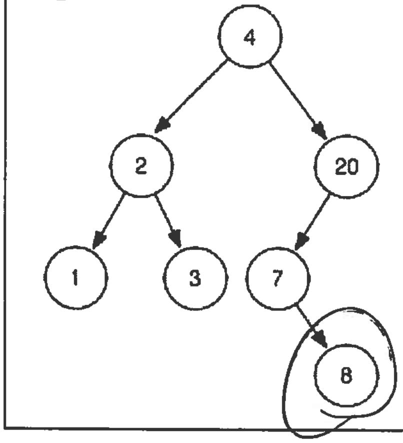
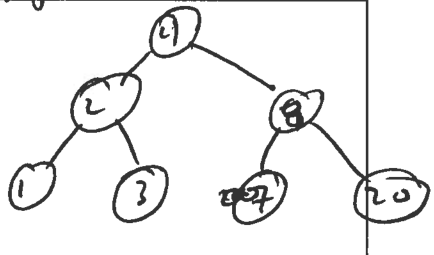
Signature: ____

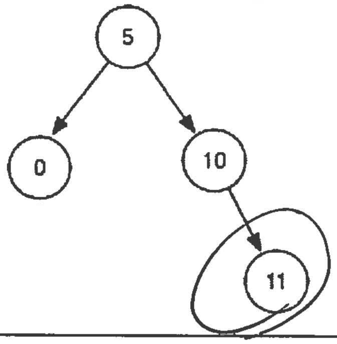
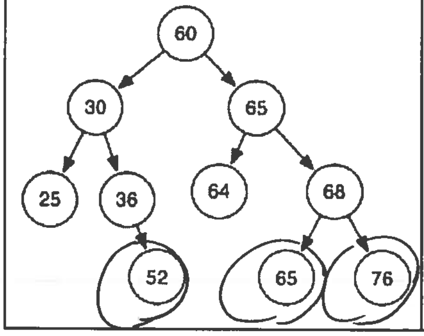
Problem	**Score**
1	/ 30
2	/ 20
3	/ 24
4	/ 26
Total	**/ 100**

Points divided evenly among parts of a problem unless otherwise specified.

Name (print): ______________________ UnityId (print) ______________@ncsu.edu

1. [30 points] In the following AVL Trees, **a single node was just inserted but the tree has not yet been balanced**. In the first column, mark the node or nodes that could have been inserted. In the second column, indicate which nodes, if any, are unbalanced ~~and their~~ balance factors. In the third column, draw the balanced tree and indicate the rotations applied. If the tree is already balanced, simply write "no rotations" and do not re-draw the tree.

AVL Tree (mark inserted node(s))	Which nodes are imbalanced?	Draw the balanced tree and list rotations applied.
	BF(25) = 2	 Rotate Right at 25
	BF(10) = -2	 right at 15 left at 10
	BF(20) = 2	 left at 7 right at 20

	None	no rotations
	None	No rotation

2. [20 points] Using the following class definition for a node, implement the following. Use valid C++ syntax (including case for keywords) to earn full points.

```
class BSTNode {
public:
   int data;
   BSTNode *left; // pointer to left sub-tree
   BSTNode *right;// pointer to right sub-tree
   BSTNode(int d, BSTNode *l=nullptr, BSTNode *r=nullptr) {
      data = d;
      left = l;
      right = r;
   }
};
```

(a) [10 points] Insertion on a Binary Search Tree using the following function prototype. See comments below for more requirements.

```
// BST_insert arguments:
// root is a reference to the root of the binary search tree. It
// could null.
//
// newNode is newly allocated and initialized node that's being
// inserted into the tree.
void BST_insert(BSTNode* &root, BSTNode* newNode);
```

```
void BST_insert ( BSTNode *&root , BSTNode * newNode ) {
      if ( root == nullptr )
         root = newNode;
      else {
           BSTNode * tmp = root;
           while ( tmp != nullptr ) {
                if ( newNode -> data < tmp -> data )
                     if ( tmp -> left == nullptr ) {
                          tmp -> left = newNode; break; }
                     else tmp = tmp -> left;

                else if ( tmp -> right == nullptr ) {
                          tmp -> right = newNode; break; }

                     else
```

```
                                        tmp = tmp->right;
                           }
                     }
             }
```

(b) [10 points] Calculate the height of a Binary Search Tree using the following function prototype.

```
// BST_height arguments:
// node is a pointer to the root of a sub-tree.
// Return the height of the sub-tree starting at node.
int BST_height(BSTNode* node);
```

```
int   BST_height (BSTNode * node) {
      if (node == nullptr) return -1;
      return 1+ std::max( BST_height(node->left), BST_height(node->right));
}
```

ECE 309 Exam 2 - Fall 2019 A

Name (print): ______________________ UnityId (print) _______________@ncsu.edu

0

1 2

3 4 5 6

3. [24 points/ 3 points each] Provide short answers for the following questions about data structures.

a. In a max-heap, what is the index for the parent of the node with index 115?

57

b. In a max-heap, what are the indices for the children of the node with index 115?

231, 232

c. What do we know about the value at the greatest valid index of a max-heap (assuming more than one value in the max-heap)?

Not the maximum, smaller than ancestors

d. What do we know about the value at the least valid index of a max-heap?

maximum value

e. In a AVL tree of integers, what's the Big-O time complexity of searching for a specific integer value in the tree?

O(log N)

f. In an AVL tree of integers, what do we know about the value at the root of the tree?

Roughly as many values bigger as smaller than it is

i.e. median

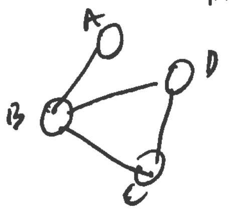
g. What's the difference between a Binary Search Tree and an AVL tree?

AVL tree is a kind of BST, nodes are ordered.

AVL is guaranteed to be balanced, search + insertion are more efficient.

e, l.

h. Give an example of an adjacency list using a small graph (4 vertices).



A → B

B → A, D, C

C → D, B

D → B, C

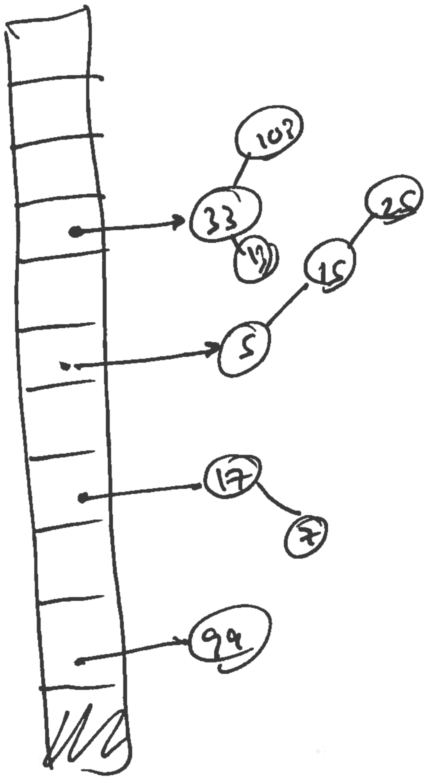
4. Consider the design of a hash table for integers in which each entry in the hash table array points to a binary search tree. For example, new items inserted in the table are inserted into the tree at the calculated hash index. Also, the hash table array is never resized, despite how many elements are inserted into the hash table. For this new data structure, answer the following questions.

(a) [2 points] Is this hash table design more like linear probing or chaining? Justify your answer.

chaining ; instead of a linked list, we use a tree to hold the chain

(b) [10 points] Assuming a table of length 10 and a hash function of key % 10, what does the hash table look like after the following insertions:

33,5,15, 25,13,103,17,7,99



(c) [7 points] Consider N to be the number of data items in the hash table and M to be the size of the table array. What is the big-O time complexity for insertion if N is much less than M assuming a perfect hash function? Justify your answer.

$O(1)$; same as chaining

(d) [7 points] Using the same definitions for N and M as in part (c) but hashing may not be perfect, what is the big-O time complexity for insertion if N is much greater than M? Justify your answer.

~~$O(\log N)$~~

$O(H)$ – height of tree

$O(\log N)$ if we use AVL tree or if data inserted is relatively random. order.

C++ Keywords

In common with C:

auto
break
case
char
const
continue
default
do
double
else
enum
extern
float
for
goto
if
int
long
register
return
short
signed
sizeof
static
struct
switch
typedef
union
unsigned
void
volatile
while

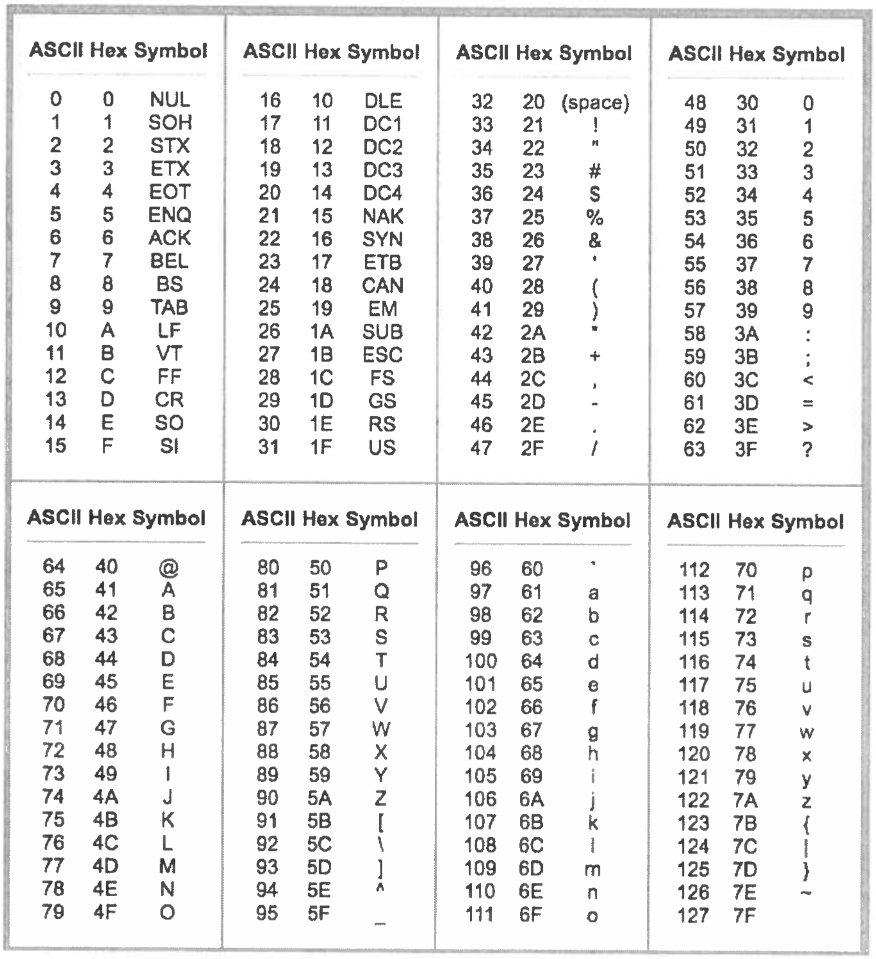
Unique to C++:

asm
bool
catch
class
const_cast
delete
dynamic_cast
explicit
false
friend
inline
mutable
namespace
new
operator
private
public
protected
reinterpret_cast
static_cast
template
this
throw
true
try
typeid
typename
using
virtual
wchar_t

Reserved words:

and
and_eq
bitand
bitor
compl
not
not_eq
or
or_eq
xor
xor_eq

ASCII Table



ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	