1 Essay 1 point Sorting - insertion sort

Sort ascending (low to high) the sequence {14, 9, 6, 3, 8, 17, 18, 5} using the insertion sort. ***Do not write code for this question!*** Show the resulting output from each step of the algorithm.

2 Multiple Choice 1 point Binary Heaps

Suppose you have an empty Min Heap and add the following values in the order shown:

8, 4, 13, 7, 17, 12, 32

You then remove the minimum value from the min heap. Which of the following is the expected output from an ***in-order traversal?***

- (●) 32, 8, 17, 7, 13, 12
- ( ) 7, 8, 32, 17, 12, 13
- ( ) 8, 17, 7, 13, 32, 12
- ( ) 7, 8, 17, 12, 13, 32
- ( ) 7, 12, 8, 17, 13, 32
- ( ) none of these

3 Fill in the Blank 1 point Min Heap

You have an empty Min heap, using the 1-based methodology discussed in class. You add the following values, one at a time, to the heap:

4, 30, 19, 16, 42, 8, 5, 6, 12

What are the values in the underlying stucture:

index 1:	4
index 2:	6
index 3:	5
index 4:	12
index 5:	42
index 6:	19
index 7:	8
index 8:	30
index 9:	16

4 Fill in the Blank 1 point Binary Heaps

Using this array:

values		4	8	7	9	3	12	5
index	0	1	2	3	4	5	6	7

Build a min heap using the 1-based methodology in the textbook (where index 0 is reserved for comparisons). Then call the deleteMinimum function on the heap twice. What is the pre-order traversal of the resulting tree?

5 , 7 , 9 ,
8 , 12

5 Fill in the Blank 1 point Hash Tables

Suppose you have a hash table with the following characteristics:

- current bucket list size is a 10 element integer array
- the hash function is (integer key MODULO bucket list size)
- collisions are resolved with quadratic probing
- the collision function is $f(i) = i^2$, where $f(0) = 0$
- rehash when load factor exceeds 0.5
- rehashing size is equal to the first prime greater than twice the current bucket list size

The following integer keys are inserted to the hash table in the order shown:

20, 15, 6, 25, 45

Show the contents of the bucket list after last element is inserted into the hash table. Enter the number or **null if empty**.

If you don't enter **null** for an empty slot it will be counted as wrong.

Array:

index 0:	20
index 1:	null
index 2:	null
index 3:	null
index 4:	45
index 5:	15
index 6:	6
index 7:	null
index 8:	null
index 9:	25

6 Numeric 1 point Hashing Load Facto

You have a hash table with the following characteristics:

- The bucket list is an array of binary search trees (BST).
- The array currently has the length of 10.
- The hash function is (integer key MODULO bucket list size).
- Collisions are resolved with separate chaining. Keys that hash to the same value are added to the BST in that bucket.

The following integer keys are inserted to the hash table in the order shown:

12, 17, 6, 5

What is the load factor (**λ**)?

0.4

7 Fill in the Blank 1 point Hash Tables Linear Probing

Suppose you have a hash table with the following characteristics:

- current bucket list size is a 10 element integer array
- the hash function is (integer key MODULO bucket list size)
- collisions are resolved with *linear probing*
- rehash when load factor exceeds 0.5
- rehashing size is equal to the first prime greater than twice the current bucket list size

The following integer keys are inserted to the hash table in the order shown:

20, 15, 6, 25, 45

Show the contents of the bucket list after last element is inserted into the hash table. Enter the number or **null if empty**.

If you don't enter **null** for an empty slot it will be counted as wrong.

Array:

index 0: 20

index 1: null

index 2: null

index 3: null

index 4: null

index 5: 15

index 6: 6

index 7: 25

index 8: 45

index 9: null

8 Fill in the Blank 1 point Double Hashing

Suppose you have a hash table with the following characteristics:

current table size is a 10 element integer array
the hash function is (integer key MODULO bucket list size)
collisions are resolved with double hashing
the collision function is f(i) = i * (R - (x MOD R)), where R is the largest prime number smaller than the table size
rehash when load factor exceeds 0.7
rehashing size is equal to the first prime greater than twice the current bucket list size

The following integer keys are inserted to the hash table in the order shown:

14, 7, 6, 27, 4, 17, 28

Show the contents of the array after last element is inserted into the hash table. Enter "null" (without the quotation marks) if the element is empty.

Array

index 0:	4
index 1:	17
index 2:	null
index 3:	null
index 4:	14
index 5:	28
index 6:	6
index 7:	7
index 8:	27
index 9:	null

9 Multiple Choice 1 point AVL Trees

After inserting the following integers in this order into an AVL tree:

10, 17, 23, 40, 12

Which integer will be the root?

- ○ 23
- ◉ 17
- ○ 10
- ○ 12

10 Multiple Choice 1 point Red-Black Trees

After inserting the following integers in this order into a Left-Leaning Red-Black tree:

10, 17, 42, 12, 50

Which integer will be the root?

- ◉ 17
- ○ 10
- ○ 42
- ○ 12

11 Multiple Choice 1 point AVL Trees 2

Inserting the following integers into an AVL tree, in the order show will require what kind of rotation(s) to maintain the AVL balance property?

12, 5, 17, 40, 42

- (●) A single left rotation
- ( ) A single right rotation
- ( ) A right-left double rotation
- ( ) A left-right double rotation
- ( ) No rotations are necessary to maintain the balance property

12 Multiple Choice 1 point AVL Trees 3

Inserting the following integers into an AVL tree, in the order show will require what kind of rotation(s) to maintain the AVL balance property?

9, 5, 16, 12, 3, 4

- ( ) A single right rotation
- ( ) A single left rotation
- ( ) A right-left double rotation
- (●) A left-right double rotation
- ( ) No rotations are necessary to maintain the balance property

13 Multiple Answer 1 point Left-Leaning Red-Black Trees 1

Entering the following integers into an Left-Leaning Red-Black tree in the order shown will result in what type(s) of binary tree?

12, 15, 42, 37, 5, 1

Select all that apply.

- [x] Complete binary tree
- [x] Balanced binary tree
- [ ] Full binary tree
- [ ] Perfect binary tree
- [ ] None of these

14 Multiple Choice 1 point Lef-Leaning Red-Black Trees

Entering the following integers into an Left-Leaning Red-Black tree in the order shown:

14, 6, 20, 4, 2

What type of rotation is necessary after inserting the last number?

- (●) A single right rotation
- ( ) A single left rotation
- ( ) A right-left double rotation
- ( ) A left-right double rotation
- ( ) No rotations are necessary

15 Multiple Choice 1 point Lef-Leaning Red-Black Trees

Entering the following integers into an Left-Leaning Red-Black tree in the order shown:

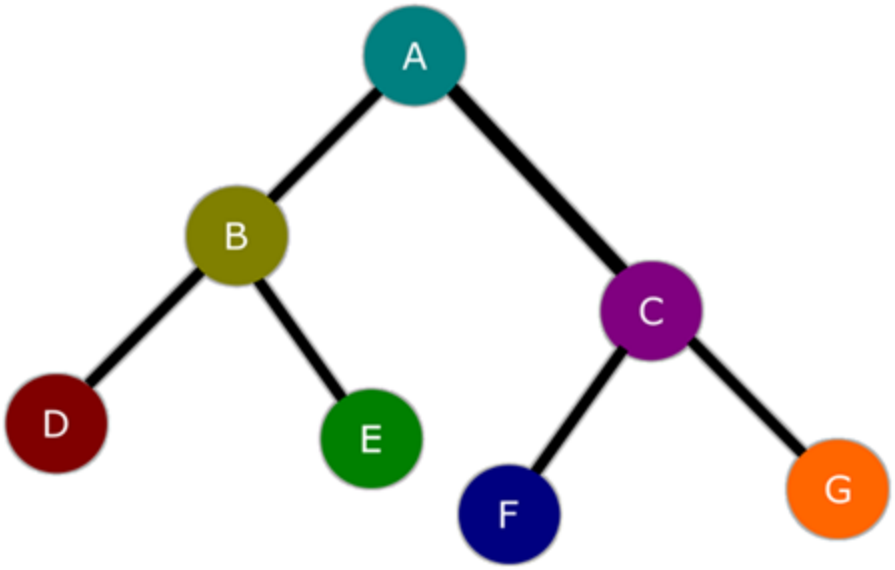
15, 8, 18, 2, 5

What type of rotation is necessary after inserting the last number?

- ○ A single right rotation
- ○ A single left rotation
- ○ A right-left double rotation
- ◉ A left-right double rotation
- ○ No rotations are necessary

16 Fill in the Blank 1 point Trees

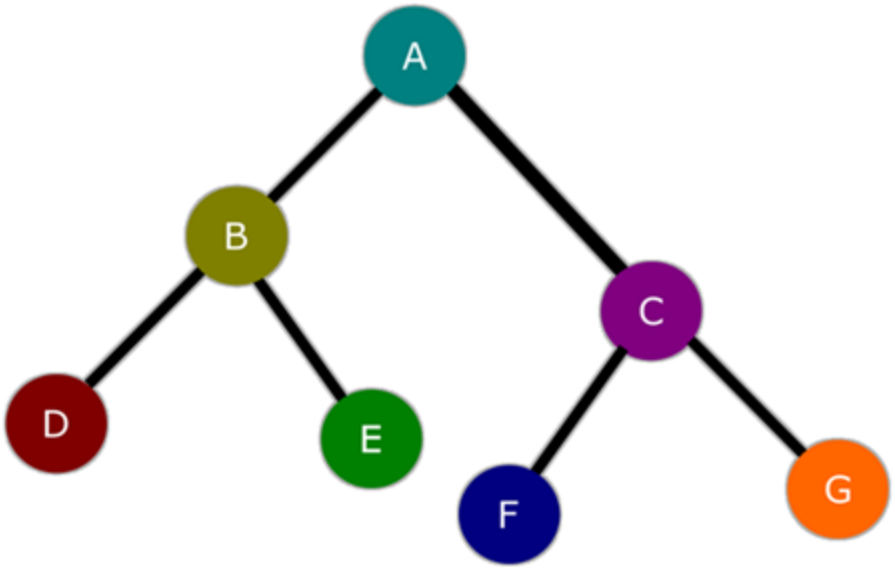
What is the *post-order* traversal of this binary tree?



D , E , B ,
F , G , C ,
A

17 Fill in the Blank 1 point Trees

What is the *in-order* traversal of this binary tree?



D , B , E ,
A , F , C ,
G

18 Multiple Answer 1 point Graphs

Given the following adjacency list for an undirected graph:

```
0:  1, 4
1:  0, 5
2:  5, 6
3:  7
4:  0, 5
5:  1, 2, 4, 7
6:  2
7:  3, 5
```

Which of the following statements are true for this graph? Select all that apply.

- [x] The graph is connected
- [ ] The graph is a tree
- [x] The path from V_6 to V_3 is V_6,V_2,V_5,V_7,V_3
- [x] The length of the path from V_1 to V_3 is 3
- [ ] The path from V_7 to V_4 is V_7,V_3,V_5,V_4

19 Multiple Answer 1 point Adacency List - Shortest Path

Given the following adjacency list:

1: 2, 3
2: 1, 3, 5
3: 1, 2, 4, 6
4: 3, 5, 6
5: 2, 4, 6, 8
6: 3, 4, 5, 7
7: 6, 8
8: 5, 7

Starting at vertex V1, determine the shortest path to all other vertices and the minimum spanning tree that these paths form. Which of these vertices are in that tree?

- [x] V1, V2
- [x] V1, V3
- [x] V2, V5
- [ ] V1, V4
- [ ] V1, V5
- [ ] V2, V3
- [x] V3, V4
- [ ] V3, V5
- [ ] V7, V8
- [x] V5, V8
- [x] V6, V3
- [ ] V3, V8
- [ ] V5, V6
- [ ] V4, V5
- [x] V6, V7

20 Essay 1 point Graph

Given the following adjacency matrix for a directed weighted graph:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0:	0	2	-	1	-	-	-	-
1:	-	-	-	-	1	-	-	-
2:	-	-	-	2	-	-	-	-
3:	-	-	-	-	1	-	5	-
4:	-	1	3	-	-	2	-	1
5:	-	-	-	2	-	-	-	3
6:	1	-	-	-	1	-	-	-
7:	-	-	2	-	-	1	-	-

Give the shortest-path from vertex 0 to each of the vertices and the cost. Give the full path **and** the cost to arrive at each vertex.

21 Fill in the Blank 1 point Graph - adjacency matrix

Given the following adjacency matrix for a directed weighted graph:

	0	**1**	**2**	**3**	**4**	**5**
0:	0	2	-	1	-	-
1:	-	-	-	-	1	-
2:	-	-	-	2	-	-
3:	-	-	-	-	1	5
4:	-	1	3	-	-	2
5:	-	-	-	2	-	-

Give the shortest-path from vertex 0 to each of the vertices and the cost. Give the full path **and** the cost to arrive at each vertex.

Use arrows without spaces between vertices to indicate path, for example, the path from a to c might like: a->b->c
Use just the number, e.g. "1" not "v1"

path to 0:	0	cost:	0
path to 1:	0->1	cost:	2
path to 2:	0->3->4->2	cost:	5
path to 3:	0->3	cost:	1
path to 4:	0->3->4	cost:	2
path to 5:	0->3->4->5	cost:	4

22 Multiple Answer 1 point Graphs Kruskal's

Define a weighted undirected graph with the following vertex and edge calls:

```
Graph g = new WeightedUndirectedGraph();
g.addVertex(0);
g.addVertex(1);
g.addVertex(2);
g.addVertex(3);
g.addVertex(4);
g.addVertex(5);
g.addVertex(6);
g.addVertex(7);

g.addEdge(0, 1, 2);
g.addEdge(0, 2, 5);
g.addEdge(1, 2, 1);
g.addEdge(1, 3, 2);
g.addEdge(2, 3, 2);
g.addEdge(2, 4, 1);
g.addEdge(2, 6, 3);
g.addEdge(3, 4, 2);
g.addEdge(3, 5, 1);
g.addEdge(4, 6, 1);
g.addEdge(4, 7, 4);
g.addEdge(5, 6, 5);
```

Create a minimal spanning tree using Kruskal's algorithm. Which edges are part of the minimal spanning tree?

- [x] v0, v1
- [ ] v0, v2
- [x] v1, v3
- [x] v1, v2
- [ ] v3, v4
- [x] v5, v3
- [ ] v2, v6
- [x] v4, v6
- [x] v4, v7
- [ ] v5, v6
- [ ] v2, v3
- [ ] v7, v1
- [ ] v3, v6

23 Fill in the Blank 1 point Disjoint Set - Path Compression

Assume the underlying data structure for a Disjoint Set has the following values:

values	1	3	3	6	5	6	-9	6	6
index	0	1	2	3	4	5	6	7	8

Assume that the ***find*** call is implemented with ***path compression***. Show the contents of the underlying data structure after the DisjointSet method, ***find(0)*** has been called.

Array

index 0: 6

index 1: 6

index 2: 3

index 3: 6

index 4: 5

index 5: 6

index 6: -9

index 7: 6

index 8: 6

24 Fill in the Blank 1 point Disjoint Arbitrary

Given a Disjoint Set which follows *union-by-arbitrary* that makes the right-hand argument subordinate to the left-hand argument, assume that the following calls have been made in the order shown:

```
DisjointSet dset = new DisjointSet( 10 );
dset.union( 1, 0 );
dset.union( 2, 3 );
dset.union( 0, 9 );
dset.union( 5, 4 );
dset.union( 4, 1 );
dset.union( 7, 8 );
dset.union( 7, 6 );
dset.union( 3, 9 );
dset.union( 1, 8 );
```

Show the contents of the underlying data structure after the DisjointSet methods have been called.

Array

index 0:	1
index 1:	5
index 2:	-1
index 3:	2
index 4:	5
index 5:	2
index 6:	7
index 7:	2
index 8:	7
index 9:	1

25 Fill in the Blank 1 point Disjoint Height

Given a Disjoint Set which follows *union-by-height* algorithm that breaks ties by making the right-hand argument subordinate to the left-hand argument, assume that the following calls have been made in the order shown:

```
DisjointSet dset = new DisjointSet( 10 );
dset.union( 1, 0 );
dset.union( 2, 3 );
dset.union( 4, 5 );
dset.union( 4, 2 );
dset.union( 7, 8 );
dset.union( 6, 7 );
dset.union( 0, 3 );
dset.union( 1, 8 );
```

Show the contents of the underlying data structure after the DisjointSet methods have been called.

Array

index 0:	1
index 1:	4
index 2:	4
index 3:	2
index 4:	-3
index 5:	4
index 6:	7
index 7:	4
index 8:	7
index 9:	-1

26 Fill in the Blank 1 point Disjoint Size

Given a Disjoint Set which follows *union-by-size with path compression algorithm* that breaks ties by making the right-hand argument subordinate to the left-hand argument, assume that the following calls have been made in the order shown:

```
DisjointSet dset = new DisjointSet( 10 );
dset.union( 1, 0 );
dset.union( 2, 3 );
dset.union( 0, 9 );
dset.union( 5, 4 );
dset.union( 4, 1 );
dset.union( 7, 8 );
dset.union( 6, 7 );
dset.union( 3, 9 );
dset.union( 8, 1 );
dset.find(8);
dset.find(4);
```

Show the contents of the underlying data structure after the DisjointSet methods have been called.

Array

index 0:	1
index 1:	-10
index 2:	1
index 3:	2
index 4:	1
index 5:	1
index 6:	7
index 7:	1
index 8:	1
index 9:	1