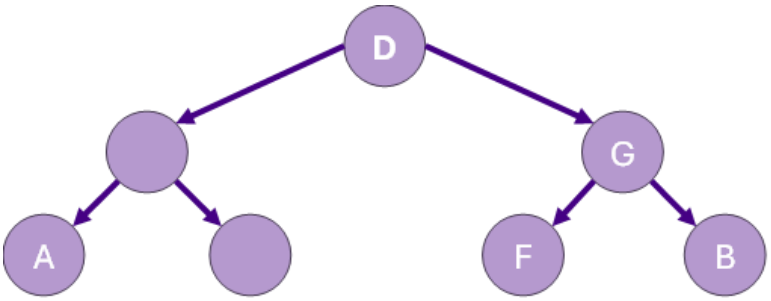
Activity: Heaps

Array Tree Representations

1. The following is a binary tree and its array representation. ***Fill in the missing values*** to make the two representations equivalent. *The root has been completed for you*.



0	1	2	3	4	5	6
D	E			C		

Heaps and Priority Queues

2. Show your work to insert the keys 21, 18, 7, 3, 45, 8 sequentially into a min-heap (in that order). ***Circle your final heap*** *and* ***write out the array representation*** *for that final heap*.

1. What is the state of the heap after a single call to `removeMin`?
   Draw both *the tree and the array representations.*

2. What is the state of the heap after a second call to `removeMin`?
   Draw both *the tree and the array representations.*

3. Here is an interface describing a simple Priority Queue ADT:

```
public interface PQueue<E extends Comparable<E>> {
  /* Enqueue the provided item. */
  void enqueue(E item);

  /* Remove and return the smallest value item. */
  E dequeue();

  /* Return, but do not remove, the smallest item. */
  E peek();

  /* Return the number of items in the PQueue. */
  int size();

  /* Return a PQueue built from the array of items. */
  PQueue<E> buildPQueue(E[] items);
}
```

   Consider if values are stored in (1) a sorted array, (2) a non-balancing BST, (3) an AVL tree, or (4) a heap.
   (Note that #1 and #4 will require a dynamic array – you may need to consider possible resizing)

   What are the **worst-case** big-Θ running times for the following operations, based on data structure?

	enqueue	**dequeue**	**peek**	**build**
1. Sorted Array				
2. BST				
3. AVL Tree				
4. Heap				