1. You have declared the following array to be used as a heap. The heap size is stored at arr[0]. Perform a bottom-up heapify, then draw the heap as a tree with nodes and edges. Circle your answer.

   ```csharp
   int arr[] = { (6), 1, 4, 2, 6, 5, 7 };   
   ```

   ![Original Tree](image1)
   ![After Heapify](image2)

2. Below is an AVL tree. Insert the value 5 then 6, in that order. Draw the final tree and circle it.

   ![AVL Tree](image3)

3. Quicksort with 3-way partitioning is especially efficient for data with what characteristic? Short answer please.

   - many duplicate values
   - few unique values
   - many repeated values

4. Huffman codes can be arranged in a binary tree because...

   A. each code leads to the next code.
   B. each code defines a unique path through the tree.
   C. each code is the prefix of another code.
   D. each code has a length equal to the tree height.

5. You must fill an integer array of size 10 with unique values that can range from 0 to 20. No duplicate values are allowed. If you use a perfect hash table to check for duplicates, what is the smallest size your hash table can be? Circle your answer.

   21
6. You have setup a hash table for quick access to data, and have written the hash function below to generate key values. Data size is $n$. A data value is passed to the hash function and a hash-key is returned. What is the asymptotic efficiency of your hash table for functions such as insert or find?

```c
int hash(char *data) {
    int val = data[0] + data[1];
    return 0;
}
```

Asymptotic efficiency of your hash table is: **Big Theta ($n$)**
A collision will occur for every operation. $n$ values will be searched.

7. How many different topological sort orders are possible with the following directed acyclic graph? There is no particular rule for decisions such as lexicographic order. Consider all topological orders. Show your work for chance for partial credit.

```
4  X Y Z W
X Y W Z
X W Y Z
W X Y Z
```

8. The source-removal algorithm for topological sorting works by removing vertices on the graph that...

A. have incoming edges  
B. have no outgoing edges  
C. have no incoming edges  
D. have incoming and outgoing edges

9. Let $S$ be a 32-element set. If one subset is generated in one operation on a computer that makes 1 billion operations per second, how long will it take to generate all subsets of $S$?  
Hint: Think about how many subsets there must be.

A. about 1/2 second  
B. around 4 seconds  
C. approximately 32 seconds  
D. a few days or longer

10. Below is a binary-search-tree, BST. The highest and lowest values in the tree are in leaf nodes. Draw your own BST in which the highest and lowest values are not in leaf nodes. Take care to draw an actual BST.

```
3 4
/ /  
2 4 1
/  
1 2
```

extra point: how many swaps were needed in the heapify of question 1? __4__