CSE 101
Final Review Problems

1. Determine whether the following statements are **True** or **False**. No justification is required.

   a. $n\sqrt{n} = \Omega(n^2)$
   b. $n^\pi = O(n^3)$
   c. $n^2 = \Theta\left(9^{\log_3(n)}\right)$
   d. $n\sqrt[3]{n} = \omega(\sqrt{n})$
   e. $n^2 = o(n^3)$
   f. $\ln(n) = o(n)$
   g. $2^n = O(n^2)$
   h. $n^{1.5} = \omega(n^{1.45})$
   i. $n\ln(n) = \Theta(\ln(\ln(n)))$
   j. $f(n) = \omega(f(n))$ for any function $f(n)$

2. Given a Binary Search Tree based on the following C++ struct

```
struct Node{
   int key;
   Node* left;
   Node* right;
};
```

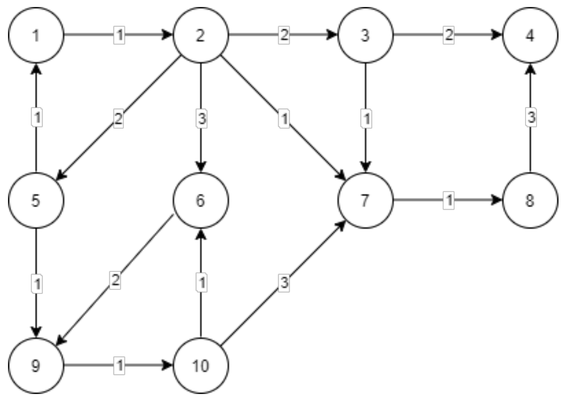
Complete the recursive C++ function below called `TreeWalk()` that takes as input a `Node` pointer `R` and a string `s`, then returns a string consisting of all keys in the subtree rooted at `R`, separated by spaces. The order of the keys depends on the input string `s`, which will be either "`pre`", "`in`" or "`post`", indicating a pre-order, in-order or a post-order tree walk, respectively. If the input `s` is not one of the strings "`pre`", "`in`" or "`post`", then your function will return the empty string. The recursion will terminate when `R` has the value `nullptr`.

```
std::string TreeWalk(Node* R, std::string s){
   // your code starts here




   // your code ends here
}
```

3. Perform Dijkstra(G, s) on the weighted digraph below with source vertex $s = 5$. If at some point two vertices have equal minimum d-values, extract the one with smaller label first from the min Priority Queue.



a. Determine the order in which vertices are extracted from the min Priority Queue.

b. For each vertex x, determine the values $d[x]$ and $p[x]$.

Solution:

x	1	2	3	4	5	6	7	8	9	10
$d[x]$										
$p[x]$										

4. Perform BuildHeap(A) on the following unordered array A, making it into a max-heap. Observe that identical keys are accompanied by letters representing different satellite data. Thus the elements 2a and 2b have the same key, but are distinguishable elements in the max-heap.

A	2a	4	7	1a	2b	3	1b	5a	2c	6	8	5b

Show the state of array A after the call to BuildHeap(A).

5. Insert the keys: 5, 9, 7, 2, 6, 4, 8, 3, 1, 10 (in order) into an initially empty Binary Search Tree T. (Note: use the Binary Search Tree Insert algorithm to do this.)

a. Give the keys in the order printed by a **pre-order tree walk**.

b. Give the keys in the order printed by a **post-order tree walk**.

Note: the three questions below **do not** refer in any way to the Red Black Tree Insert algorithm. Instead they ask if it is possible to assign colors in the BST *T*, which you found above, so as to satisfy the RBT properties. Be sure to include nil children when computing the black-height of *T*.

c. Is it possible to assign the colors $\{\text{Red}, \text{Black}\}$ to the vertices of T so that the Red-Black Tree properties are satisfied, and $\text{bh}(T) = 1$? If it is possible, specify all such colorings by stating, for each coloring, the set of keys belonging to red nodes.

d. Is it possible to assign the colors $\{\text{Red}, \text{Black}\}$ to the vertices of T so that the Red-Black Tree properties are satisfied, and $\text{bh}(T) = 2$? If it is possible, specify all such colorings by stating, for each coloring, the set of keys belonging to red nodes.

e. Is it possible to assign the colors $\{\text{Red}, \text{Black}\}$ to the vertices of T so that the Red-Black Tree properties are satisfied, and $\text{bh}(T) = 3$? If it is possible, specify all such colorings by stating, for each coloring, the set of keys belonging to red nodes.