

CSE 101

Final Review Problems

1. Determine whether the following statements are **True** or **False**.

- a. $n\sqrt{n} = \Omega(n^2)$
- b. $n^\pi = O(n^3)$
- c. $n^2 = \Theta(9^{\log_3(n)})$
- d. $n^3\sqrt{n} = \omega(\sqrt{n})$
- 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 $f(n)$
- k. $\ln(\ln(n)) = o(\ln(n))$
- l. $\ln(n) = \omega(n)$
- m. $n^2 = O(n^3)$
- n. $\pi^n = O(n^2)$
- o. $2^{\sqrt{n}} = o(3^n)$
- p. $5^n = \Omega(n^5)$
- q. $2 + \sin(n) = \Theta(1)$
- r. $n^{1.4} = \Theta(n^{1.45})$

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)
```

3. Given a connected (undirected) graph G , and a vertex $x \in V(G)$, the *eccentricity* of x in G is the maximum possible distance from x to any other vertex $y \in V(G)$, i.e.

$$\text{eccentricity}(x) = \max\{ \delta(x, y) \mid y \in V(G) \}.$$

The *diameter* of a graph is the maximum eccentricity over all vertices in the graph, i.e.

$$\text{diameter}(G) = \max\{ \text{eccentricity}(x) \mid x \in V(G) \}.$$

- a. Using only the Graph ADT functions defined in the project description for pa2, write a client function in C with the heading

```
int eccentricity(Graph G, int x)
```

that returns the eccentricity of x , for any x in the range $1 \leq x \leq |V(G)|$. This function need not check the precondition $1 \leq x \leq |V(G)|$.

- b. Using only the Graph ADT functions defined in the project description for pa2, and the function `eccentricity()` from part (a), write a client function in C with the heading

```
int diameter(Graph G)
```

that returns the diameter of G . This function has no preconditions.

6. 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.)
- Give the keys in the order printed by a **pre-order tree walk**.
 - 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 .

- Is it possible to assign the colors {Red, 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.
- Is it possible to assign the colors {Red, 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.
- Is it possible to assign the colors {Red, 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.

7. Insert the keys: 4, 2, 8, 1, 6, 10, 7, 3, 12, 5, 13, 9, 11 (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 of T in the order printed by a **pre-order tree walk**.
- b. Delete the keys 4 and 8 from T (in order). Give the keys of T in the order printed by a **post-order tree walk**.
- c. Perform a Left-Rotation about the root of the tree T resulting from part (b). Give the keys of T in the order printed by a **pre-order tree walk**.

8. Insert the keys: 6, 2, 1, 4, 3, 5 (in order) into an initially empty Red-Black Tree (using the `RB_Insert()` algorithm), then draw the resulting tree T . Indicate the color of each node in T .

9. Insert the keys: 5, 1, 2, 3, 4 (in order) into an initially empty Red-Black Tree (using the `RB_Insert()` algorithm), then draw the resulting tree T . Indicate the color of each node in T .

10. Perform $\text{BuildHeap}(A)$ on the following unordered array, 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 , and the corresponding tree picture, after the call to $\text{BuildHeap}(A)$.