نام و نام خانوادگي	پارميدا فهندژ
نام درس	ساختمان داده

سؤال ۲:
درخت دودویی کامل (Complete Binary Tree): درختی است که در آن همه‌ی سطوح به‌جز آخرین سطح، کاملاً پر هستند و گره‌های سطح آخر از سمت چپ به راست پر شده‌اند.

درخت دودویی کامل (Full Binary Tree): درختی است که هر گره یا هیچ فرزندی ندارد (برگ است) یا دقیقاً دو فرزند دارد.

مثال نقض:
درختی با ریشه و تنها یک فرزند چپ یا راست، یک درخت دودویی کامل نیست (نیست Full)، ولی می‌تواند در شرایطی جزئی از یک درخت دودویی کامل باشد.

سؤال ۳:
جمله‌ی غلط:

> "در پیمایش inorder، همیشه مقدار گره‌ها افزایشی (n از ۱ تا n) مرتب می‌شوند."

این جمله فقط در مورد درخت جست‌وجوی دودویی (BST) درست است، نه همه‌ی درخت‌های دودویی.

سؤال ۴:
برای پیدا کردن LCA (Lowest Common Ancestor) دو گره در BST:

* اگر مقدار گره فعلی بین مقدارهای A و B باشد، همان گره LCA است.
* اگر هر دو مقدار کمتر از گره فعلی باشند، به زیردرخت چپ بروید.
* اگر هر دو مقدار بیشتر باشند، به زیردرخت راست بروید.

سؤال ۵:

داده شده. دو روش برای جست‌وجو (key) و یک کلید (BST) یک اشاره‌گر به ریشه یک درخت جست‌وجوی دودویی:

۱. **بازگشتی (Recursive):**
```
def search(root, key):
    if root is None or root.val == key:
        return root
    if key < root.val:
        return search(root.left, key)
    return search(root.right, key)
```

۲. **تکراری (Iterative):**
```
def search(root, key):
    while root is not None and root.val != key:
        if key < root.val:
            root = root.left
        else:
            root = root.right
    return root
```

سؤال ۶:
در یک درخت دودویی (leaf nodes) تابعی برای شمارش تعداد برگ‌ها:
```
def count_leaves(root):
    if root is None:
        return 0
    if root.left is None and root.right is None:
        return 1
    return count_leaves(root.left) + count_leaves(root.right)
```

این تابع با استفاده از بازگشت، گره‌هایی که فرزند ندارند (برگ‌ها) را شمرده و جمع می‌زند.

سؤال ۷:

محاسبه تعداد گره‌ها در یک درخت دودویی:

روش بازگشتی (recursive):

```
def count_nodes(root):
    if root is None:
        return 0
    return 1 + count_nodes(root.left) + count_nodes(root.right)
```

روش تکراری (iterative):

با استفاده از صف (queue):

```
from collections import deque

def count_nodes(root):
    if root is None:
        return 0
    queue = deque([root])
    count = 0
    while queue:
        node = queue.popleft()
        count += 1
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return count
```