DS2030 DSA for DS
Quiz 1: 50 minutes

Name:
Roll Number:

Instructions

Answer all the questions below. Write your answers directly on the quiz paper in the spaces provided. The total marks for the quiz is 30 points. Allocate your time wisely. *Describe the algorithms using the pseudo-code notation used in the class or Python syntax.You are not allowed to use Python list operations unless explicitly stated.*

1. (2 points) Write a Python code snippet to remove all even numbers from a **Python list**.

   Solution:

```
    nums = [num for num in nums if num % 2 != 0]
```

2. (4 points) Describe an algorithm to find the middle element of a singly linked list in one pass assuming that the singly linked list class does not have a length attribute.

   Solution:

```
    def find_middle(head):
        slow = head
        fast = head
        while fast and fast.next:
            slow = slow.next
            fast = fast.next.next
        return slow.value
```

3. (2 points) What is the time complexity of the following `find_min` function? Justify your answer.

```
    def find_min(arr):
        min_val = arr[0]
        for num in arr:
            if num < min_val:
                min_val = num
        return min_val
```

Solution:

$O(n)$

4. (2 points) Give an example of a scenario where using a stack would be more appropriate than using a queue. Explain your reasoning.

   Solution:

   Undo operation in an editor requiring LIFO operation.

   Recursion also requiring LIFO operation

5. (2 points) Write an algorithm function that uses a stack to check if a given string is a palindrome. A palindrome is a string that reads the same forward and backward. For example, "adam", "racecar", and "level" are palindromes, while "hello" and "world" are not. You can assume that the check is case-insensitive and all the functions of a Stack are already defined.

   Solution:

```
    def is_palindrome(s):
        # Convert string to lowercase to make the check case-insensitive. This is
            optional.
        s = s.lower()

        # Initialize a stack
        stack = Stack()

        # Push all filtered characters to the stack
        for char in s:
            stack.push(char)

        # Check for palindrome by popping from the stack
        for char in s:
            if char != stack.pop():
                return False

        return True
```

6. (6 points) Write the algorithm that merges two *sorted* singly linked lists into one sorted singly linked list. Your function should take the heads of two *sorted* linked lists as input and return the head of the merged sorted list. The algorithm must use the existing nodes in the linked lists and not create new nodes.

Solution:

```
def merge_sorted_lists(l1, l2):
    # Create a dummy node to simplify the merging process
    dummy = ListNode(0)
    current = dummy

    # Traverse both lists and append the smaller value to the merged list
    while l1 and l2:
        if l1.value < l2.value:
            current.next = l1
            l1 = l1.next
        else:
            current.next = l2
            l2 = l2.next
        current = current.next

    # Append the remaining nodes of l1 or l2
    current.next = l1 if l1 else l2

    # Return the merged list, starting from the node after the dummy node
    return dummy.next
```

This function uses a dummy node to simplify edge cases. It iteratively compares the nodes from both lists and appends the smaller one to the result list. The remaining nodes of the non-exhausted list are appended at the end.

7. (6 points) Prove that $T(n) = 3n^2 + 5n + 2$ is $\Theta(n^2)$.

Solution: To prove that $T(n) = 3n^2 + 5n + 2$ is $\Theta(n^2)$, we need to find constants c_1, c_2, and n_0 such that:

$$c_1 \cdot n^2 \leq 3n^2 + 5n + 2 \leq c_2 \cdot n^2 \quad \text{for all } n \geq n_0.$$

Upper Bound (O Notation): To prove the upper bound, we want to show that there exists a constant $c_2 > 0$ and $n_0 > 0$ such that:

$$3n^2 + 5n + 2 \leq c_2 \cdot n^2 \quad \text{for all } n \geq n_0.$$

For large n, the n^2 term dominates the $5n$ and 2 terms. To find c_2, we can factor out n^2:

$$3n^2 + 5n + 2 \leq n^2(3 + \frac{5}{n} + \frac{2}{n^2}).$$

As n becomes large, $\frac{5}{n} \to 0$ and $\frac{2}{n^2} \to 0$, so:

$$3n^2 + 5n + 2 \leq 4n^2 \quad \text{(choosing } c_2 = 4).$$

Thus, for $c_2 = 4$, we can say that:

$$3n^2 + 5n + 2 \leq 4n^2 \quad \text{for all } n \geq n_0.$$

To find n_0, we solve the inequality $n^2 \geq 5n + 2$:

$$n^2 - 5n - 2 \geq 0.$$

We can estimate n_0 by using the quadratic formula $n = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ for $a = 1, b = -5, c = -2$. The roots are approximately $n \approx 5.4$. Thus, $n_0 = 6$ works, ensuring $n \geq 6$.

Lower Bound (Ω Notation): To prove the lower bound, we need to show that there exists a constant $c_1 > 0$ and $n_0 > 0$ such that:

$$c_1 \cdot n^2 \leq 3n^2 + 5n + 2 \quad \text{for all } n \geq n_0.$$

Again, for large n, the n^2 term dominates. We choose $c_1 = 2$ such that:

$$2n^2 \leq 3n^2 + 5n + 2.$$

Rearranging the inequality:

$$n^2 \leq 5n + 2.$$

This is true for $n \geq 1$. Thus, $n_0 = 1$ works for $c_1 = 2$.

Thus, with $c_1 = 2, c_2 = 4$, and $n_0 = 6$, we have shown that:

$$2n^2 \leq 3n^2 + 5n + 2 \leq 4n^2 \quad \text{for all } n \geq 6.$$

Therefore, $T(n) = 3n^2 + 5n + 2 = \Theta(n^2)$.

8. (6 points) Given a binary tree, write the algorithm to find the **Lowest Common Ancestor (LCA)** of two given nodes in the tree. The LCA of two nodes p and q in a binary tree is defined as the deepest node that has both p and q as descendants (where we allow a node to be a descendant of itself).

Solution: Here is the Python function 'findLCA' using a recursive approach to find the Lowest Common Ancestor in a binary tree:

```
        def findLCA(root, p, q):
            if not root:
                return None

            # If the current node is either p or q
            if root == p or root == q:
                return root

            # Recur for left and right subtrees
            left = findLCA(root.left, p, q)
            right = findLCA(root.right, p, q)

            # If p and q found in left and right subtrees of the   current node
            if left and right:
                return root

            # Otherwise, check if left subtree or right subtree is LCA
            return left if left else right
```

The function works as follows:

(a) **Base Case**: If the current 'root' node is 'None', return 'None'.

(b) If the 'root' is either 'p' or 'q', return the 'root'.

(c) Recursively find 'p' and 'q' in the left and right subtrees.

(d) If both 'p' and 'q' are found in different subtrees of the current node, then the current node is the LCA.

(e) If only one subtree contains both nodes, return that subtree's result.

Time Complexity Analysis:

- The time complexity of this approach is $O(n)$, where n is the number of nodes in the binary tree, as each node is processed once.