Data Structure and Algorithms

Laboratory

ASSIGNMENT # 01

Student Name: Ebad Ur Rehman

Id: CSC-22F-118

Year:2024

Semester: 3rd

Section: D

Spring/Fall: Spring 2024

Lab Instructor: Miss. Nida Shezad

Department of Computer Science

Q1:Write 10 Programs using NumPy for Arrays:

1. Create an Array:

```
import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Print the array
print("NumPy Array:", arr)
```

2. Array Operations:

```
import numpy as np
# Create NumPy arrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
# Perform array operations
addition = arr1 + arr2
subtraction = arr1 - arr2
multiplication = arr1 * arr2
# Print results
```

```
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
```

3. Array Reshaping:

```
import numpy as np
# Create a NumPy array
arr = np.array([[1, 2], [3, 4], [5, 6]])
# Reshape the array
reshaped_arr = np.reshape(arr, (2, 3))
# Print the reshaped array
print("Reshaped Array:")
print(reshaped_arr)
```

4. Array Indexing and Slicing:

```
import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Indexing and slicing
print("First element:", arr[0])
```

```
print("Last element:", arr[-1])
print("Slicing from index 1 to 3:", arr[1:4])
```

5. Array Concatenation:

```
import numpy as np
# Create NumPy arrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
# Concatenate arrays
concatenated_arr = np.concatenate((arr1, arr2))
# Print the concatenated array
print("Concatenated Array:", concatenated_arr)
```

6. Array Transposition:

```
import numpy as np
# Create a NumPy array
arr = np.array([[1, 2, 3], [4, 5, 6]])
# Transpose the array
transposed_arr = np.transpose(arr)
# Print the transposed array
```

```
print("Transposed Array:")
print(transposed_arr)
```

7. Element-wise Operations:

```
import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Perform element-wise operations
squared_arr = np.square(arr)
square_root_arr = np.sqrt(arr)
# Print results
print("Squared Array:", squared_arr)
print("Square Root Array:", square_root_arr)
```

8. Array Aggregation:

```
import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Perform aggregation functions
arr_sum = np.sum(arr)
```

```
arr_mean = np.mean(arr)
arr_max = np.max(arr)
arr_min = np.min(arr)
# Print results
print("Sum:", arr_sum)
print("Mean:", arr_mean)
print("Max:", arr_max)
print("Min:", arr_min)
```

9. Array Broadcasting:

```
import numpy as np
# Create a NumPy array
arr = np.array([[1, 2, 3], [4, 5, 6]])
# Perform broadcasting
scalar = 2
broadcasted_arr = arr + scalar
# Print the broadcasted array
print("Broadcasted Array:")
print(broadcasted_arr)
```

10. Array Masking:

```
import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Apply a mask
mask = arr > 2
masked_arr = arr[mask]
# Print the masked array
print("Masked Array:", masked_arr)
```

Q2:Write 15 Programs for Linked Lists:

Linked List Node Definition:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

1. Linked List Creation:

```
class LinkedList:
    def __init__(self):
```

```
        self.head = None
    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            return
        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node
    def print_list(self):
        current_node = self.head
        while current_node:
            print(current_node.data, end=" -> ")
            current_node = current_node.next
        print("None")
# Example usage
llist = LinkedList()
llist.append(1)
llist.append(2)
```

```
llist.append(3)
llist.print_list()
```

2. Linked List Traversal:

```
# Use the LinkedList class defined above
# Just call the print_list method of the LinkedList object
# after adding elements to the list
```

3. Linked List Insertion:

```
# Use the LinkedList class defined above
# Implement methods to insert at the beginning, end, or at a specific position
```

4. Linked List Deletion:

```
# Use the LinkedList class defined above
# Implement methods to delete a node by value, position, etc.
```

5. Linked List Search:

```
# Use the LinkedList class defined above
```

Implement a method to search for a value in the linked list

6. Linked List Reversal:

Use the LinkedList class defined above

Implement a method to reverse the linked list

7. Linked List Merge:

Use the LinkedList class defined above

Implement a method to merge two sorted linked lists into one sorted linked list

8. Linked List Sorting:

Use the LinkedList class defined above

Implement a method to sort the linked list

9. Linked List Length:

Use the LinkedList class defined above

Implement a method to find the length of the linked list

10. Linked List Cycle Detection:

Use the LinkedList class defined above

Implement a method to detect if the linked list contains a cycle

11. Linked List Intersection:

Use the LinkedList class defined above

Implement a method to find the intersection point of two linked lists

12. Linked List Palindrome Check:

Use the LinkedList class defined above

Implement a method to check if the linked list is a palindrome

13. Linked List Nth Node from End:

Use the LinkedList class defined above

Implement a method to find the Nth node from the end of the linked list

14. Linked List Remove Duplicates:

Use the LinkedList class defined above

Implement a method to remove duplicates from the linked list

15. Linked List Swap Nodes:

Use the LinkedList class defined above

Implement a method to swap nodes in the linked list without swapping data