Rojin and Suresh Assignment 2

1. Pseudocode

removeDuplicates(arr):

```
resultSet = emptySet  // To store unique elements
result = emptyArray  // To store the final result

for each num in arr:
    if num not in resultSet:
        resultSet.add(num)
        result.push(num)

return result
```

```
export function remDupe<T>(arr: T[]): T[] {
  let resultSet = new Set<T>();
  let result: T[] = [];
  for (const num of arr) {
    if (!resultSet.has(num)) {
      resultSet.add(num);
      result.push(num);
    }
  }

  return result;
}
```

2. Pseudocode

isPermutation(str1, str2):

```
if length(str1) ≠ length(str2):
    return false  // Different lengths cannot be permutations

for i = 0 to length(str1) - 1:
    if str1[i] ≠ str2[length(str2) - 1 - i]:
        return false  // Mismatch found

return true  // Strings are permutations of each other
```

```
export function isPermutation(str1: string, str2: string): boolean {
  if (str1.length !== str2.length) return false;

  for (let i = 0; i < str1.length; i++) {
```

```
  if (str1[i] !== str2[str2.length - 1 - i]) return false;
}


return true;}
```

Referencs:

```
Algorithm sum(L)
  Input: Doubly Linked List
  Output: Sum of all the elements in the given list
    if L.isEmpty() then return 0
    p := L.first() // Get the first node
    sum := p.element() // get the value present in the first node
    while ! L.isLast(p) do // loop through until reach the last node
              p := L.after(p) // Move the next node after p
              sum := sum + p.element() // get the value present in the p node
     return sum // after the loop, return the computed sum
```

3. DLL Practice with List ADT.

   A. Implement the function to get the maximum value from the given Doubly Linked list.

```
Algorithm findMax(L)
  Input: Doubly Linked List L
   Output: Maximum value from the given DLL
     If L.isEmpty() then return null
     a = L.first()
     maxVal = a .element() // assuming max is the value from first node
    while a.next() is not null do
     if a.next().element() > maxVal then // comparing
        maxVal = a.next().element() // new max
       a = L.after(a) // moves to the next node
   return maxVal
```

Time Complexity is O(n).

   B. Implement the function to return the middle of Double linked list. For a implementation hint refer Slide 23.

```
Algorithm findMiddle(L)
  Input: Doubly Linked List L
   Output: Middle elemenet from the given DLL
     If L.isEmpty() then return null
      leftPointer = L.first() // first node
```

RightPointer = L.right() // last node
While leftPointer != rightPointer and leftPointer.next() != rightPointer do
leftPointer = L.after(leftPointer) // move leftPointer to the next node
rightPointer = L.before(rightPointer) // move rightPointer to the backward node

return leftPointer.element() // return the elemet where two pointers meet each other

Time Complexity is O(n).

C. Implement the function to remove the middle element in the given list.
Hint: You can use the implemented function of Task B in the Task C.
Algorithm removeMiddle(L)
Input: Doubly Link List L
Output: None

middleElement = findMiddle(L) // use method from task B
if middleElement is null then return // nothing to remove
prev = L.before(middleElement) // get the node before the middle element
next = L.after(middleElement) // get the node after the middle element

if prev is not null then
prev.next = next // adjust its next pointer
if next is not null then
next.prev = prev // adjusting its previous pointer
else
L.head = next //remove the head node
if next is not null then // if there is a node after the head, adjust its previous pointer
next.prev = null