Microsoft DSA Practice

1. Finding the second largest number in an array

   Solution:

```
class Solution {
  public int getSecondLargest(int[] arr) {
    // code here
    Arrays.sort(arr);
    int n = arr.length;
    for(int i=n-1;i>=0;i--){
      if(i-1>=0 && arr[i]!=arr[i-1]) return arr[i-1];
    }
    return -1;
  }
}
```

   Time Complexity: o(nlogn)
   Space: o(1);

   Solution 2:

```
class Solution {
  public int getSecondLargest(int[] arr) {
    // code here
    int max1 = Integer.MIN_VALUE;
    int n = arr.length;
    for(int i=0;i<n;i++){
      if(arr[i]>max1) max1= arr[i];
    }
    int max2 = Integer.MIN_VALUE;
    for(int i=0;i<n;i++){
      if(arr[i]>max2 && arr[i]<max1) max2 = Math.max(max2,arr[i]);
    }
    return max2==Integer.MIN_VALUE?-1:max2;
  }
}
```

Time Complexity: o(n)
Space Complexity: o(1)

2. Reverse a Linked List:

   Solution:

```
class Solution {
  public ListNode reverseList(ListNode head) {
    if(head==null) return null;
    ListNode curr = head;
    ListNode prev = null;
    ListNode prevnode = new ListNode(head.val);
    ListNode heada = prevnode;
    while(curr!=null){
      if(prev!=null){
        ListNode newnode = new ListNode(curr.val);
        newnode.next = prevnode;
        heada = newnode;
        prevnode = newnode;
      }
      curr = curr.next;
      prev = curr;
    }
    return heada;
  }
}
```

   Time Complexity: o(n)
   Space Complexity: o(n)

3. Replace space with %20 in a string

   Solution:

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Codechef
{
public static void main (String[] args) throws java.lang.Exception
{
// your code goes here
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
  String[] arr = s.split(" ");
  StringBuilder sb = new StringBuilder();
  for(int i=0;i<arr.length;i++){
    sb.append(arr[i]);
    if(i!=arr.length-1) sb.append("%20");
  }
  System.out.println(sb.toString());

}
}
```

Time Complexity: o(n)
Space Complexity: o(n)

4. Product of array without itself

   Solution:

```
class Solution {
  public int[] productExceptSelf(int[] nums) {
    int n = nums.length;
    int[] pref = new int[n];
    int[] suff = new int[n];
    int prod = 1;
    for(int i=0;i<n;i++){
      prod *= nums[i];
      pref[i] = prod;
    }
```

```
        prod = 1;
        for(int i=n-1;i>=0;i--){
          prod *= nums[i];
          suff[i] = prod;
        }
        int[] ans = new int[n];
        for(int i=0;i<n;i++){
          int prev = (i-1>=0)?pref[i-1]:1;
          int next = (i+1<n)?suff[i+1]:1;
          ans[i] = prev*next;
        }
        return ans;
      }
    }
```

Time Complexity: o(n)
Space Complexity: o(n)

Note: Used pref and suff product since they said we have to solve this without division operator

5. Splitting the phone number, age and the seat no

   Solution:

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Codechef
{
public static void main (String[] args) throws java.lang.Exception
{
// your code goes here
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
int n = s.length();
StringBuilder phone = new StringBuilder();
```

```
    char gender = 'M';
    StringBuilder age = new StringBuilder();
    StringBuilder seat = new StringBuilder();
    for(int i=0;i<n;i++){
      if(i<10){
        phone.append(s.charAt(i));
      }else if(i==10){
        gender = s.charAt(i);
      }else if(i<n-2){
        age.append(s.charAt(i));
      }
    }
    System.out.println("Phone "+phone.toString());
    System.out.println("Gender "+gender);
    System.out.println("Age "+age);
    System.out.println("Seat no "+s.substring(n-2,n));
    if(Integer.parseInt(age.toString())>50) System.out.println("He is the senior
citizen");
    else System.out.println("He is not a senior citizen");

}
}
```

Time complexity: o(n)
Space complexity: o(n)

6. Find the missing Number

Solution:

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Codechef
{
public static void main (String[] args) throws java.lang.Exception
{
```

```
// your code goes here
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
Set<Integer> set = new HashSet<>();
int[] arr = new int[n];

for(int i=0;i<n-1;i++){
   arr[i] = sc.nextInt();
   set.add(arr[i]);
}

for(int i=1;i<=n;i++){
   if(!set.contains(i)){
      System.out.println("Missing number is "+i);
      return;
   }
}
System.out.println("All numbers are present");


}
}
```

Time Complexity: o(n)

Space Complexity: o(n-1)

7. Merge Intervals

   Solution:

```
class Solution {

   public int[][] merge(int[][] inter) {

      Arrays.sort(inter, (a,b)->{

         return Integer.compare(a[0],b[0]);

      });

      List<int[]> ans = new ArrayList<>();

      int n = inter.length;
```

```
        int i = 0;
        int start = inter[i][0];
        int end = inter[i][1];
        while(i<n){
          start = inter[i][0];
          end = inter[i][1];
          while(i+1<n && end>=inter[i+1][0]){
            end = Math.max(end,inter[i+1][1]);
            i++;
          }
          ans.add(new int[]{start,end});
          i++;
        }
        int size = ans.size();
        int[][] fin = new int[size][2];
        for(int j=0;j<size;j++){
          fin[j] = new int[]{ans.get(j)[0],ans.get(j)[1]};
        }
        return fin;
      }
    }
```

Time complexity: o(n)

Space complexity: o(n)

8. Merge two linked lists

Solution:

```
class Solution {

public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

ListNode ans = new ListNode(0);

ListNode dummy = ans;

while(list1!=null && list2!=null){

if(list1.val<list2.val){

ans.next = new ListNode(list1.val);

list1 = list1.next;

}else{

ans.next = new ListNode(list2.val);

list2 = list2.next;

}

ans = ans.next;

}

while(list1!=null){

ans.next = new ListNode(list1.val);

list1 = list1.next;

ans = ans.next;

}

while(list2!=null){

ans.next = new ListNode(list2.val);
list2 = list2.next;

ans = ans.next;

}

return dummy.next;

}

}
```

Time Complexity: o(n)

Space Complexity: o(n)

9. Getting the Employee with the second highest salary

   Solution:

```
import java.util.*;

class Employee {
  String name;
  int age;
  double salary;

  Employee(String name, int age, double salary) {
    this.name = name;
    this.age = age;
    this.salary = salary;
  }
}

public class Main {
  public static void main(String[] args) {
    List<Employee> employees = Arrays.asList(
      new Employee("Arun", 30, 50000),
            new Employee("Bhuvi", 25, 60000),
            new Employee("Kumar", 28, 55000),
            new Employee("Ravi", 32, 60000)
        );

        employees.sort((a, b) -> Double.compare(b.salary, a.salary));
        double highest = employees.get(0).salary;
        for (Employee e : employees) {
            if (e.salary < highest) {
                System.out.println(e.name);
                break;
            }
        }
    }
}
```

Time complexity: o(nlogn)
Space complexity: o(n)