Worksheet - 8

Ques 1 : https://leetcode.com/problems/happy-number/

Code :

```
class Solution{
   public boolean isHappy(int n) {
           Set<Integer> set = new HashSet<>();
       while(true){
                int sum=0;
          while(n!=0){
             sum += Math.pow(n%10,2.0);
             n = n/10;
          }
                  if(sum ==1){
             return true;
          }
           n= sum;
           if(set.contains(n)){
                 return false;
           }
           set.add(n);
           }
      }
}
```

Explaination :

HashSet to detect when the number sum is already present in set or not

find the sum of squares by using the Math.pow method. After calculating the sum of squares check if sum is equal to 1 then return true otherwise put a number equal to the sum and check if the set contains the number n or not. if yes return false otherwise add number n in the set repeat the process until the sum is equal to 1 .

Input : 31

Output: true

Input : 2

Output : false

Ques 2 : https://leetcode.com/problems/add-two-numbers/description/

Code :

```
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode dummy = new ListNode(0);
        ListNode temp = dummy;
        int carry = 0;
        while(l1 != null || l2 != null){
            int sum = 0;
            if(l1 != null){
                sum += l1.val;
                l1 = l1.next;
            }
            if(l2 != null){
                sum += l2.val;
                l2 = l2.next;
            }
            sum += carry;
            carry = sum / 10;
            temp.next = new ListNode(sum % 10);
            temp = temp.next;
        }
        if(carry != 0){
            temp.next = new ListNode(carry);
        }
        return dummy.next;
    }
}
```

Explanation :

Creating a dummy node is used to build the result linked list. Let the other node at head of dummy ListNode is temp. Variables sum and carry initializes with 0. While loop that runs when l1 or l2 are null. If conditions when l1 is not equal to null then add l1 node value to the sum and traverse to the next node. Similar to the l2 node.calculate carry. If carry is

not equal to 0 then a new node is created and put the carry value in the next node. Lastly, return the dummy node.

Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7, 0, 8]

Ques 3: https://leetcode.com/problems/two-sum/description/

Code :

```
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int n = nums.length;
        int i = 0;
        int j = i+1;
        while(i<j){
            if(nums[i] + nums[j] == target){
                return new int[] {i,j};
            }
            else if(j == n-1){
                i++;
                j = i+1;
            }else{
                j++;
            }
        }
        return new int[] {-1,-1};
    }
}
```

Explanation :

Take two variables i and j. i initialize with 0 and j initialize with i+1, loop will be go i to j check conditions with nested if else :

1. nums[i] + nums[j] = target then return new array with index i and j
2. J == n-1 , then increment i with 1 and put j = i+1
3. else increment j with 1

Lastly return the array with i j value {-1 -1} when target not match

Input: nums=[2, 7, 11, 15] target = 9
Output : [0, 1]

Ques 4: https://leetcode.com/problems/palindrome-number/

Code :

```
class Solution {
   public boolean isPalindrome(int x) {
      String s=Integer.toString(x);
      int n=s.length();
      for(int i=0;i<n/2;i++){
         if(s.charAt(i)!=s.charAt(n-i-1)){
            return false;
         }
      }
      return true;
   }
}
```

Explanation : take a string s convert to the integer using Integer.toString(), traverse through for loop that goes 0 to n/2 and check condition : s.charAt(i) is not equal to s.charAt(n-i-1),(string reads from left to right and right to left would not be same) return false otherwise return true,(string reads from left to right and right to left would not be same).

Input: x = 121
Output: true
Input: x = -121
Output: false

Ques 5: https://leetcode.com/problems/same-tree/description/

Code :

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
```

```
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public  boolean isSameTree(TreeNode p, TreeNode q) {
                if (p == null && q == null) {
                    return true;
                }
                if (p == null || q == null) {
                    return false;
                }
                if (p.val != q.val) {
                    return false;
                }
        return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
    }
}
```

Explanation:

Base case: if both nodes are null, they are equal
If one of the nodes is null and the other is not, they are not equal
Check if the current nodes have the same value
Recursively check the left and right subtrees

Input: p = [1,2,3], q = [1,2,3]

Output: true

Ques 6: https://leetcode.com/problems/merge-sorted-array/

Code:

```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int[] res = new int[m+n];
        int i=0,j=0,k=0;
        while(i<m && j<n){
            if(nums1[i]<nums2[j]){
                res[k]=nums1[i];
                k++;i++;
            }
```

```
            else{
                res[k]=nums2[j];
                k++;j++;
            }
        }
        while(i<m){
            res[k]=nums1[i];
            i++;k++;
        }
        while(j<n){
            res[k]=nums2[j];
            j++;k++;
        }
        for(i=0;i<(m+n);i++){
            nums1[i]=res[i];
        }
    }
}
```

Explanation:

The arrays we are merging are nums1 and nums2.

Create a new array res with length n+m

The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

Output: [1,2,2,3,5,6]

Ques 7: https://leetcode.com/problems/reverse-integer/

Code:

```
class Solution {
    public int reverse(int x) {
        int reverse =0;
        while(x!=0){
            int n=x%10;
            if(reverse<Integer.MIN_VALUE/10 || reverse>Integer.MAX_VALUE/10){
                return 0;
            }
            reverse=reverse*10+n;
            x=x/10;
        }
```

```
        return reverse;
    }
}
```

Explanation:

Take a variable reverse to store the reverse integer value. While loop for traverse to identify the reverse is minimum or maximum integer then return 0, otherwise calculate the value of reverse by reverse*10 +n formula when loop ends return reverse.

Input : 123

Output : 321