Assignment 9 (Recursion)

Question 1

Given an integer `n`, return *`true` if it is a power of two. Otherwise, return `false`*.

An integer `n` is a power of two, if there exists an integer `x` such that `n == 2x`.

Solution:

```
 public boolean isPowerOfTwo(int n) {
     if (((n<=0)) || (Integer.lowestOneBit(n) != n)) {
         return false;
         }
        else {
         return true;
        }
    }
```

Question 2

Given a number n, find the sum of the first natural numbers.

Solution:

```
 public int missingNumber(int[] nums) {
     int n = nums.length + 1;
        int total = (n * (n-1 )) / 2;

        for (int num : nums) {
            total -= num;
        }

        return total;
    }
```

Question 3

Given a positive integer, N. Find the factorial of N.

Solution:

```
 public int clumsy(int n) {
        int ans = 1;
        if(n <= 4){
            if(n <= 2) return n;
            else if(n == 3) return 6;
            else if(n == 4) return 7;
        } else {
            if(n%4 == 1 || n%4 == 2) ans = n+2;
            else if(n%4 == 3) ans = n-1;
            else ans = n+1;
        }
        return ans;
    }
```

Question 4

Given a number N and a power P, the task is to find the exponent of this number raised to the given power, i.e. N^P.

Solution:

```
 public double myPow(double x, int n) {
        return Math.pow(x,n);
    }
```

Question 5

Given an array of integers **arr**, the task is to find maximum element of that array using recursion.

Solution:

```
public static int findMaxRec(int A[], int n)
  {
    // if size = 0 means whole array
    // has been traversed
    if(n == 1)
      return A[0];


      return Math.max(A[n-1], findMaxRec(A, n-1));
  }
```

Question 6

Given first term (a), common difference (d) and a integer N of the Arithmetic Progression series, the task is to find Nth term of the series.

Solution:

```
public boolean canMakeArithmeticProgression(int[] arr) {
        Arrays.sort(arr);
        int diff = arr[1] - arr[0];


        for (int i = 2; i < arr.length; i++) {
            if (arr[i] - arr[i-1] != diff) {
                return false;
            }
        }
```

```
        return true;
    }
```

Question 7

Given a string S, the task is to write a program to print all permutations of a given string.

Solution:

```
 public boolean checkInclusion(String s1, String s2) {
        if(s2.length() < s1.length()){
            return false;
        }
        int p1 = 0;
        int p2 = 0;

        HashMap<Character, Integer> map = new HashMap<>();
        for(int i = 0 ; i < s1.length(); i++){
            if(map.containsKey(s1.charAt(i))){
                map.put(s1.charAt(i), map.get(s1.charAt(i))+1);
            }
            else{
                map.put(s1.charAt(i), 1);
            }
        }
        // we now know the size of the window
```

```
HashMap<Character, Integer> map2 = new HashMap<>();
ArrayList<Character> list = new ArrayList<>();
for(int i = 0 ; i < s1.length(); i++){
    list.add(s2.charAt(i));
    if(map2.containsKey(s2.charAt(i))){
        map2.put(s2.charAt(i), map2.get(s2.charAt(i))+1);
    }
    else{
        map2.put(s2.charAt(i), 1);
    }
}
if(map2.equals(map)){
    return true;
}
for(int i = s1.length(); i < s2.length(); i++){
    char ch = list.get(0);
    list.remove(0);
    list.add(s2.charAt(i));
    map2.put(ch, map2.get(ch)-1);
    if(map2.get(ch) == 0){
        map2.remove(ch);
    }
    if(map2.containsKey(s2.charAt(i))){
```

```
            map2.put(s2.charAt(i), map2.get(s2.charAt(i))+1);
        }
        else{
            map2.put(s2.charAt(i), 1);
        }
        if(map2.equals(map)){
            return true;
        }
      }
      return false;
  }
```

Question 8

Given an array, find a product of all array elements.

Solution:

```
 public int[] productExceptSelf(int[] nums) {
    int [] sm = new int[nums.length];
    int [] pm = new int[nums.length];
    int [] ans = new int[nums.length];
    pm[0] = nums[0];
    sm[nums.length-1] = nums[nums.length-1];

    for(int i = 1; i<nums.length; i++){
      pm[i] = pm[i-1]*nums[i];
```

```
}

for(int i =nums.length-2; i>=0; i--)
{
    sm[i] = sm[i+1]*nums[i];
}

for(int i =0; i<nums.length; i++)
{
    if(i==0)
     ans[i] = 1*sm[i+1];
    else if(i==nums.length-1)
    ans[i] = pm[i-1]*1;
    else
    ans[i] = sm[i+1]*pm[i-1];
}

return ans; }
```