Worksheet- 21

507. Perfect Number

```
class Solution {
public boolean checkPerfectNumber(int num) {
// Initialize the sum variable to 1
int sum = 1;
// Check if the number is 1
if(num==1) {
// If the number is 1, it is not a perfect number, so return false
return false;
}
// Iterate from 2 up to the square root of the number
for(int i=2; i*i<=num; i++) {
// Check if the number is divisible by i
if(num%i==0) {
// If divisible, add i and num/i to the sum
sum += i + num/i;
}
}
// Check if the sum is equal to the original number
if(sum==num) {
// If the sum is equal to the number, it is a perfect number, so return true
return true;
}
// If the sum is not equal to the number, it is not a perfect number, so return false
return false;
}
}
```

```
public class Main {
public static void main(String[] args) {
// Create an instance of the Solution class
Solution solution = new Solution();
// Test number 6
int num1 = 6;
System.out.println(num1 + " is a perfect number: " + solution.checkPerfectNumber(num1));
// Test number 28
int num2 = 28;
System.out.println(num2 + " is a perfect number: " + solution.checkPerfectNumber(num2));
// Test number 12
int num3 = 12;
System.out.println(num3 + " is a perfect number: " + solution.checkPerfectNumber(num3));
// Test number 8
int num4 = 8;
System.out.println(num4 + " is a perfect number: " + solution.checkPerfectNumber(num4));
}
}
```

420. Strong Password Checker

```
public int strongPasswordChecker(String s) {
// Initialize counters for lowercase letters, uppercase letters, and digits
int res = 0, a = 1, A = 1, d = 1;
// Convert the input string to a character array
char[] carr = s.toCharArray();
// Initialize an array to store the consecutive repetition counts of characters
int[] arr = new int[carr.length];

// Iterate over the character array
for (int i = 0; i < arr.length;) {
// Set lowercase counter to 0 if a lowercase letter is found
if (Character.isLowerCase(carr[i])) a = 0;
// Set uppercase counter to 0 if an uppercase letter is found
if (Character.isUpperCase(carr[i])) A = 0;
// Set digit counter to 0 if a digit is found
if (Character.isDigit(carr[i])) d = 0;

int j = i;
// Count the consecutive repetition of a character
while (i < carr.length && carr[i] == carr[j]) i++;
// Store the repetition count in the array
arr[j] = i - j;
}
// Calculate the total number of missing character types (lowercase, uppercase, and digit)
int total_missing = (a + A + d);
// If the password length is less than 6
if (arr.length < 6) {
// Add the missing character types and additional characters required to meet the minimum length
res += total_missing + Math.max(0, 6 - (arr.length + total_missing));
```

```
// If the password length is 6 or greater
} else {
// Calculate the excess length of the password and initialize variables for additional changes
int over_len = Math.max(arr.length - 20, 0), left_over = 0;
// Add the excess length to the result
res += over_len;
// Iterate twice to consider different possible modifications
for (int k = 1; k < 3; k++) {
for (int i = 0; i < arr.length && over_len > 0; i++) {

// Skip the repetition counts that do not require modification
if (arr[i] < 3 | | arr[i] % 3 != (k - 1)) continue;
// Reduce the repetition count by the required modifications
arr[i] -= Math.min(over_len, k);
// Reduce the excess length accordingly
over_len -= k;
}
}

for (int i = 0; i < arr.length; i++) {
if (arr[i] >= 3 && over_len > 0) {
// Calculate the additional modifications required to reduce repetition count to 2
int need = arr[i] - 2;
// Reduce the repetition count by available modifications
arr[i] -= over_len;
// Reduce the excess length accordingly
over_len -= need;
}
// Count the remaining repetition counts that are greater than or equal to 3
if (arr[i] >= 3) left_over += arr[i] / 3;
}
```

```
// Add the missing character types or the remaining repetition counts to the result
res += Math.max(total_missing, left_over);
}
// Return the final result
return res;
}

public class Main {
public static void main(String[] args) {
// Test password
String password = "MyPassword123";
// Create an instance of the Solution class
Solution solution = new Solution();
int result = solution.strongPasswordChecker(password);
System.out.println(result);
}
}
```