# Q1. https://leetcode.com/problems/happy-number/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 18/05/2023  
 \*/*public class QuestionOneSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*isHappy*(19));  
 System.*out*.println(*isHappy*(32));  
 System.*out*.println(*isHappy*(15));  
 }  
  
 private static boolean isHappy(int n) {  
 if (n == 1 || n == 7) return true;  
 if (n < 10) return false;  
 int sum = 0;  
 while (n != 0) {  
 sum += (n % 10) \* (n % 10);  
 n /= 10;  
 }  
 return *isHappy*(sum);  
 }  
}

# Q2. https://leetcode.com/problems/power-of-two/

## Solution*:*

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 18/05/2023  
 \*/*public class QuestionTwoSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*powerOfTwo*(16));  
 System.*out*.println(*powerOfTwo*(3));  
 }  
  
 static boolean powerOfTwo(int n) {  
 return (n > 0) && ((n & (n - 1)) == 0);  
 }  
}

# Q3. https://leetcode.com/problems/valid-anagram/

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 18/05/2023  
 \*/*public class QuestionThreeSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*isAnagramMethod1*("anagram", "nagaram"));  
 System.*out*.println(*isAnagramMethod2*("anagram", "nagaram"));  
 }  
  
 *//time complexity -> O(nlogn)* static boolean isAnagramMethod1(String s, String t) {  
 char[] sArray = s.toCharArray();  
 Arrays.*sort*(sArray);  
 char[] tArray = t.toCharArray();  
 Arrays.*sort*(tArray);  
 return Arrays.*equals*(sArray, tArray);  
 }  
  
 *//time complexity -> O(n), space complexity -> O(1)* static boolean isAnagramMethod2(String s, String t) {  
 int[] freq = new int[26];  
 for(char c: s.toCharArray()){  
 freq[c-'a']++;  
 }  
 for(char c:t.toCharArray()){  
 freq[c-'a']--;  
 }  
 return Arrays.*stream*(freq).allMatch(x->x==0);  
 }  
}

# Q4. https://leetcode.com/problems/ugly-number/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 18/05/2023  
 \*/*public class QuestionFourSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*isUgly*(6));  
 System.*out*.println(*isUgly*(14));  
 System.*out*.println(*isUgly*(8));  
 }  
  
 *//time complexity -> O(logn)* static boolean isUgly(int n) {  
 if (n == 0) return false;  
 for (int i = 2; i <= 5; i += i - 1) {  
 while (n % i == 0) {  
 n /= i;  
 }  
 }  
 return n == 1;  
 }  
}

# Q5. https://leetcode.com/problems/reverse-vowels-of-a-string/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/05/2023  
 \*/*public class QuestionFiveSolution {  
 public static void main(String[] args) {  
 QuestionFiveSolution solution = new QuestionFiveSolution();  
 System.*out*.println(solution.reverseVowels("Hello"));  
 System.*out*.println(solution.reverseVowels("Leetcode"));  
 }  
  
 public String reverseVowels(String s) {  
 int i = 0;  
 int j = s.length() - 1;  
 char[] cArray = s.toCharArray();  
 while (i < j) {  
 if (!isVowel(cArray[i])) {  
 i++;  
 continue;  
 }  
 if (!isVowel(cArray[j])) {  
 j--;  
 continue;  
 }  
 char temp = cArray[i];  
 cArray[i] = cArray[j];  
 cArray[j] = temp;  
  
 i++; j--;  
 }  
 return new String(cArray);  
 }  
  
 public boolean isVowel(char c) {  
 return (c == 'a' || c == 'A' ||  
 c == 'e' || c == 'E' ||  
 c == 'i' || c == 'I' ||  
 c == 'o' || c == 'O' ||  
 c == 'u' || c == 'U'  
 );  
 }  
}

# Q6. https://leetcode.com/problems/third-maximum-number/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/05/2023  
 \*/*public class QuestionSixSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findThirdMax*(new int[]{3, 2, 1}));  
 }  
  
 static int findThirdMax(int[] nums) {  
 long max = Long.*MIN\_VALUE*;  
 long secondMax = Long.*MIN\_VALUE*;  
 long thirdMax = Long.*MIN\_VALUE*;  
  
 for (int num : nums) {  
 if (num > max) {  
 thirdMax = secondMax;  
 secondMax = max;  
 max = num;  
 } else if (num > secondMax && num < max) {  
 thirdMax = secondMax;  
 secondMax = num;  
 } else if (num > thirdMax && num < secondMax) {  
 thirdMax = num;  
 }  
 }  
 return (thirdMax == Long.*MIN\_VALUE*) ? (int) max : (int) thirdMax;  
 }  
}

# Q7. https://leetcode.com/problems/find-the-difference/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/05/2023  
 \*/*public class QuestionSevenSolution {  
 public static void main(String[] args) {  
 QuestionSevenSolution solution =new QuestionSevenSolution();  
 System.*out*.println(solution.findTheDifference("abcd","abecd"));  
 }  
  
 public char findTheDifference(String s, String t) {  
 char c = 0;  
 for (char ch : s.toCharArray()) {  
 c ^= ch;  
 }  
 for (char ch : t.toCharArray()) {  
 c ^= ch;  
 }  
 return c;  
 }  
}

# Q8. https://leetcode.com/problems/add-digits/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/05/2023  
 \*/*public class QuestionEightSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*addDigits*(38));  
 System.*out*.println(*addDigits*(0));  
 }  
  
 static int addDigits(int num) {  
 if (num == 0) return 0;  
 return (num % 9 == 0) ? 9 : num % 9;  
 }  
}

# Q9. https://leetcode.com/problems/sum-of-digits-of-string-after-convert/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/05/2023  
 \*/*public class QuestionNineSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*getLucky*("iiii",1));  
 System.*out*.println(*getLucky*("leetcode",2));  
 System.*out*.println(*getLucky*("zbax",3));  
 }  
  
 public static int getLucky(String s, int k) {  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < s.length(); i++) {  
 sb.append((s.charAt(i) - 'a') + 1);  
 }  
  
 String result = sb.toString();  
 if (result.length() == 1) {  
 return Character.*getNumericValue*(result.charAt(0));  
 }  
  
 int sum = 0;  
 while (k > 0 && result.length() > 1) {  
 sum = 0;  
 for (int i = 0; i < result.length(); i++)  
 sum += Character.*getNumericValue*(result.charAt(i));  
  
 result = String.*valueOf*(sum);  
 k--;  
 }  
 return sum;  
 }  
}