Ali Abdullah

L1f24bsse0332

T1:

public class TwinPrimes {  
  
 public static void main(String[] args) {  
 for (int i = 3; i < 1000 - 2; i++) {  
 if (*isPrime*(i) && *isPrime*(i + 2)) {  
 System.*out*.println("(" + i + ", " + (i + 2) + ")");  
 }  
 }  
 }  
  
 public static boolean isPrime(int number) {  
 if (number <= 1) return false;  
 for (int i = 2; i <= Math.*sqrt*(number); i++) {  
 if (number % i == 0) return false;  
 }  
 return true;  
 }  
}

T2:

import java.util.Random;  
  
public class CrapsGame {  
  
 public static void main(String[] args) {  
 int sum = *rollDice*();  
  
 if (sum == 2 || sum == 3 || sum == 12) {  
 System.*out*.println("You lose");  
 } else if (sum == 7 || sum == 11) {  
 System.*out*.println("You win");  
 } else {  
 int point = sum;  
 System.*out*.println("point is " + point);  
  
 while (true) {  
 sum = *rollDice*();  
 if (sum == point) {  
 System.*out*.println("You win");  
 break;  
 } else if (sum == 7) {  
 System.*out*.println("You lose");  
 break;  
 }  
 }  
 }  
 }  
  
 public static int rollDice() {  
 Random rand = new Random();  
 int die1 = rand.nextInt(6) + 1;  
 int die2 = rand.nextInt(6) + 1;  
 int sum = die1 + die2;  
 System.*out*.println("You rolled " + die1 + " + " + die2 + " = " + sum);  
 return sum;  
 }  
}

T3:

public class Encryptor {  
  
 public static int encryptNumber(int n) {  
 int result = 0;  
 int multiplier = 1;  
  
 while (n > 0) {  
 int digit = n % 10;  
 int square = digit \* digit;  
  
 int temp = square;  
 int digits = 1;  
 while (temp >= 10) {  
 temp /= 10;  
 digits \*= 10;  
 }  
  
 result = result \* digits + square;  
 n /= 10;  
 }  
  
 int reversed = 0;  
 while (result > 0) {  
 reversed = reversed \* 10 + result % 10;  
 result /= 10;  
 }  
  
 return reversed;  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println(*encryptNumber*(321));  
 System.*out*.println(*encryptNumber*(705));  
 }  
}

T4:

public class AlternatingPatternChecker {  
  
 public static boolean isAlternatingPattern(int n) {  
 int[] digits = *getDigits*(n);  
 if (digits.length < 2) return false;  
  
 Boolean goingUp = *getInitialDirection*(digits);  
 if (goingUp == null) return false;  
  
 for (int i = 1; i < digits.length - 1; i++) {  
 if (goingUp && !(digits[i] > digits[i - 1] && digits[i] > digits[i + 1])) {  
 return false;  
 } else if (!goingUp && !(digits[i] < digits[i - 1] && digits[i] < digits[i + 1])) {  
 return false;  
 }  
 goingUp = !goingUp;  
 }  
  
 return true;  
 }  
  
 public static int[] getDigits(int n) {  
 String s = String.*valueOf*(Math.*abs*(n));   
 int[] digits = new int[s.length()];  
 for (int i = 0; i < s.length(); i++) {  
 digits[i] = s.charAt(i) - '0';  
 }  
 return digits;  
 }  
  
 public static Boolean getInitialDirection(int[] digits) {  
 if (digits.length < 2) return null;  
 if (digits[1] > digits[0]) return true;  
 if (digits[1] < digits[0]) return false;  
 return null;  
 }  
  
 public static boolean isPeak(int a, int b, int c) {  
 return b > a && b > c;  
 }  
  
 public static boolean isValley(int a, int b, int c) {  
 return b < a && b < c;  
 }  
  
 public static void main(String[] args) {  
 int[] testCases = {2745, 1234, 8652, 214365, 121, 2323};  
 for (int num : testCases) {  
 System.*out*.println(num + " -> " + *isAlternatingPattern*(num));  
 }  
 }  
}

T5:

public class MagicCodeGenerator {  
  
 public static int generateMagicCode(int power) {  
 int reversedPower = *reverseNumber*(power);  
 int result = 0;  
  
 while (reversedPower > 0) {  
 int digit = reversedPower % 10;  
 int square = digit \* digit;  
 result = result \* (int)Math.*pow*(10, *getDigitCount*(square)) + square;  
 reversedPower /= 10;  
 }  
  
 return *reverseNumber*(result);  
 }  
  
 public static int reverseNumber(int n) {  
 int reversed = 0;  
 while (n > 0) {  
 reversed = reversed \* 10 + n % 10;  
 n /= 10;  
 }  
 return reversed;  
 }  
  
 public static int getDigitCount(int n) {  
 if (n == 0) return 1;  
 int count = 0;  
 while (n > 0) {  
 count++;  
 n /= 10;  
 }  
 return count;  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println(*generateMagicCode*(512)); // Expected: 4152  
 System.*out*.println(*generateMagicCode*(90)); // Expected: 81  
 System.*out*.println(*generateMagicCode*(0)); // Expected: 0  
 System.*out*.println(*generateMagicCode*(123)); // 1^2=1, 2^2=4, 3^2=9 -> "149" -> reversed: 941  
 }  
}

T6:

import java.util.\*;  
  
public class SmartCalendarAssistant {  
  
 public static boolean isLeapYear(int year) {  
 return (year % 4 == 0) && ((year % 100 != 0) || (year % 400 == 0));  
 }  
  
 public static int getDaysInMonth(int month, int year) {  
 switch (month) {  
 case 2:  
 return *isLeapYear*(year) ? 29 : 28;  
 case 4: case 6: case 9: case 11:  
 return 30;  
 default:  
 return 31;  
 }  
 }  
  
 public static String getDayOfWeek(int day, int month, int year) {  
 if (month < 3) {  
 month += 12;  
 year--;  
 }  
 int q = day;  
 int m = month;  
 int k = year % 100;  
 int j = year / 100;  
  
 int h = (q + 13 \* (m + 1) / 5 + k + k / 4 + j / 4 + 5 \* j) % 7;  
 String[] days = {"Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"};  
 return days[h];  
 }  
  
 public static boolean isWeekend(int day, int month, int year) {  
 String dayOfWeek = *getDayOfWeek*(day, month, year);  
 return dayOfWeek.equals("Saturday") || dayOfWeek.equals("Sunday");  
 }  
  
 public static String compareDates(String date1, String date2) {  
 int[] d1 = *parseDate*(date1);  
 int[] d2 = *parseDate*(date2);  
  
 Calendar cal1 = new GregorianCalendar(d1[2], d1[1] - 1, d1[0]);  
 Calendar cal2 = new GregorianCalendar(d2[2], d2[1] - 1, d2[0]);  
  
 if (cal1.equals(cal2)) return "Both dates are the same";  
 else if (cal1.before(cal2)) return "Date1 is earlier";  
 else return "Date2 is earlier";  
 }  
  
 private static int[] parseDate(String date) {  
 String[] parts = date.split("-");  
 return new int[] {  
 Integer.*parseInt*(parts[0]),  
 Integer.*parseInt*(parts[1]),  
 Integer.*parseInt*(parts[2])  
 };  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println(*isLeapYear*(2024)); // true  
 System.*out*.println(*getDaysInMonth*(2, 2024)); // 29  
 System.*out*.println(*getDayOfWeek*(25, 12, 2024)); // "Wednesday"  
 System.*out*.println(*isWeekend*(25, 12, 2024)); // false  
 System.*out*.println(*compareDates*("25-12-2024", "25-12-2024")); // same  
  
 System.*out*.println(*isLeapYear*(2025)); // false  
 System.*out*.println(*getDaysInMonth*(2, 2025)); // 28  
 System.*out*.println(*getDayOfWeek*(1, 1, 2025)); // "Wednesday"  
 System.*out*.println(*isWeekend*(1, 1, 2025)); // false  
 System.*out*.println(*compareDates*("01-01-2025", "31-12-2024")); // "Date2 is earlier"  
 }  
}

T7:

public class ChequeAmountInWords {  
  
 private static final String[] *units* = {  
 "", "One", "Two", "Three", "Four", "Five",  
 "Six", "Seven", "Eight", "Nine", "Ten", "Eleven",  
 "Twelve", "Thirteen", "Fourteen", "Fifteen",  
 "Sixteen", "Seventeen", "Eighteen", "Nineteen"  
 };  
  
 private static final String[] *tens* = {  
 "", "", "Twenty", "Thirty", "Forty", "Fifty",  
 "Sixty", "Seventy", "Eighty", "Ninety"  
 };  
  
 public static String convertToWords(int number) {  
 if (number == 0) {  
 return "Zero";  
 }  
  
 StringBuilder result = new StringBuilder();  
  
 result.append(*convertSegment*(number / 10000000, "Crore"));  
 result.append(*convertSegment*((number / 100000) % 100, "Lakh"));  
 result.append(*convertSegment*((number / 1000) % 100, "Thousand"));  
 result.append(*convertSegment*((number / 100) % 10, "Hundred"));  
  
 int lastTwoDigits = number % 100;  
 if (number > 100 && lastTwoDigits > 0) {  
 result.append(" and ");  
 }  
  
 result.append(*convertBelowHundred*(lastTwoDigits));  
  
 String finalResult = result.toString().trim();  
 return Character.*toUpperCase*(finalResult.charAt(0)) + finalResult.substring(1);  
 }  
  
 private static String convertSegment(int num, String label) {  
 if (num == 0) return "";  
 return *convertBelowHundred*(num) + " " + label + " ";  
 }  
  
 private static String convertBelowHundred(int num) {  
 if (num < 20) {  
 return *units*[num];  
 } else {  
 return *tens*[num / 10] + (num % 10 != 0 ? " " + *units*[num % 10] : "");  
 }  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println(*convertToWords*(0)); // Zero  
 System.*out*.println(*convertToWords*(1542)); // One Thousand Five Hundred Forty Two  
 System.*out*.println(*convertToWords*(1002003)); // Ten Lakh Two Thousand Three  
 System.*out*.println(*convertToWords*(999999999)); // Ninety Nine Crore Ninety Nine Lakh Ninety Nine Thousand Nine Hundred Ninety Nine  
 System.*out*.println(*convertToWords*(500001)); // Five Lakh One  
 System.*out*.println(*convertToWords*(100)); // One Hundred  
 }  
}