**Лабораторная работа 2. Простые типы данных**

**1. Найдите НОК (наименьшее общее кратное) двух натуральных чисел.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Два натуральных числа **a** и **b** (**a**, **b** < **2** \* **109**). | Вывести НОК чисел **a** и **b**. |
| 42 24 | 168 |

import java.util.Scanner;

public class Lab2\_1 {

    public static int nod(int a, int b){

        return (b == 0) ? a : nod(b, a % b);

    }

    public static int nok(int a, int b){

        return a \* b / nod(a,b);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int a = sc.nextInt();

        int b = sc.nextInt();

        sc.close();

        System.out.println(nok(a,b));

    }

}



**2. Найти НОД (наибольший общий делитель) n чисел.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Первая строка содержит количество чисел **n** (**1** < **n** < **101**). Во второй строке через пробел заданы **n** натуральных чисел, каждое из которых не превышает **30000**. | НОД заданных чисел. |
| 2  15 25 | 5 |

import java.util.Scanner;

public class Lab2\_2 {

    public static int nod(int a, int b){

        return (b == 0) ? a : nod(b, a % b);

    }

    public static int NodNumbers(int[] numbers){

        int result = numbers[0];

        for(int i = 0; i < numbers.length; i++)

            result = nod(result, numbers[i]);

        return result;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int[] numbers = new int[n];

        for(int i = 0; i < n; i++)

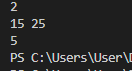
            numbers[i] = sc.nextInt();

        sc.close();

        System.out.println(NodNumbers(numbers));

    }

}



**3. Числа Фибоначчи - это последовательность чисел F(n), которая задаётся формулой:**

**F(0) = 1, F(1) = 1, F(n) = F(n - 1) + F(n - 2)**

**По заданному числу n вывести n-ое число Фибоначчи.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Неотрицательное число **n** (**n** ≤ **45**) - номер числа Фибоначчи, которое нужно вывести. | Вывести **n**-ое число Фибоначчи. |
| 4 | 5 |

import java.util.Scanner;

public class Lab2\_3{

    public static int Fib(int n){

            return n == 0 ? 1 : (n == 1) ? n : Fib(n-1) + Fib(n-2);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        sc.close();

        System.out.println(Fib(n));

    }

}



**4. Последовательность града образуется следующим образом:**

* **Если n четно, то делим его на 2 и присваиваем n**
* **Если n нечетно, то умножим его на 3, прибавим 1 и присваиваем n**

**Утверждается, что для любого натурального числа n указанная последовательность всегда заканчивается циклом: 4, 2, 1, 4, 2, 1, ... Достаточно сказать, что при n = 1 последовательность заканчивается.**

**Напишите программу, которая определит наибольшее значение в последовательности для заданного числа n.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Первая строка содержит количество тестов **t** (**1** ≤ **t** ≤ **100000**). Каждый тест следует обработать независимо от других.  Каждый тест состоит из одной строки, содержащей два целых числа. Первое число указывает на номер теста. Вторым является число **n** (**1** ≤ **n** ≤ **100000**) - начальное число последовательности. | Для каждого теста вывести в отдельной строке его номер, пробел, и наибольшее число, встречающееся во всей последовательности начиная и включительно с **n**. |
| 4  1 1  2 3  3 9999  4 100000 | 1 1  2 16  3 101248  4 100000 |

import java.util.Scanner;

public class Lab2\_4 {

    public static int grad(int n){

        int maxValue = n;

       while(n != 1){

        if(n > maxValue)

        maxValue = n;

        if(n % 2 == 0)

            n /= 2;

        else

            n = n \* 3 + 1;

       } return maxValue;

    }

    public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);

        int t = sc.nextInt();

        for(int i =0; i < t; i++){

            System.out.print(i+1 + " ");

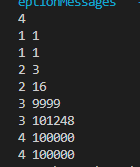
            int n = sc.nextInt();

            System.out.println(i+1 + " " + grad(n));

        } sc.close();

    }

}



**5. Даны n чисел. Найдите среди них число с минимальной суммой цифр (а из чисел с одинаковой суммой – последнее).**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| В первой строчке находится число **n** (**1** ≤ **n** ≤ **105**). В следующей строке через пробел перечислены все числа. Все числа натуральные, не превышающие **109**. | Выведите одно число – искомый минимум. |
| 4  7 8 9 10 | 10 |
| 4  41 32 23 14 | 14 |

import java.util.Scanner;

public class Lab2\_5 {

    public static int minSumNumber(int[] numbers){

        int minSum = 99999999;

        int minNumber = numbers[0];

        for(int i = 0; i < numbers.length; ++i){

            if(minSum >= NumberSum(numbers[i])){

                minSum = NumberSum(numbers[i]);

                minNumber = numbers[i];

            }

        }

        return minNumber;

    }

    public static int NumberSum(int number){

        int sum = 0;

        while(number > 0){

            sum += number % 10;

            number /= 10;

        }

        return sum;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int[] numbers = new int[n];

        for(int i = 0; i < n; ++i) {

            numbers[i] = sc.nextInt();

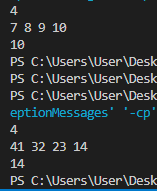
        }

        System.out.println(minSumNumber(numbers));

        sc.close();

    }

}



**6. Сколько натуральных чисел из промежутка [m, n] имеют наименьшую сумму цифр?**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Два натуральных числа **m** и **n** (**1** ≤ **m** ≤ **n** ≤ **106**). | Вывести количество натуральных чисел из промежутка [**m**, **n**] с наименьшей суммой цифр. |
| 1 100 | 3 |

import java.util.Scanner;

public class Lab2\_6 {

    public static int NumberSum(int number){

        int sum = 0;

        while(number > 0){

            sum += number % 10;

            number /= 10;

        } return sum;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int m = sc.nextInt();

        int n = sc.nextInt();

        sc.close();

        int minSum = 99999, count = 0;

        for(int i = m; i <= n; ++i){

            if(minSum >= NumberSum(i)){

                minSum = NumberSum(i);

                if(minSum == NumberSum(i))

                ++count;

            }

        }

        System.out.println(count);

    }

}



**7. По трем натуральным числам x, n и m вычислить значение xn mod m.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Три натуральных числа **x**, **n**, **m** (**1** ≤ **x**, **m** ≤ **109**, **2** ≤ **n** ≤ **107**). | Вывести **xn** mod **m**. |
| 2 3 100 | 8 |

import java.util.Scanner;

public class Lab2\_7 {

    public static int Pow(int x, int n){

        int result = 1;

        for(int i = 1; i <= n; ++i){

            result \*= x;

        } return result;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int x = sc.nextInt();

        int n = sc.nextInt();

        int m = sc.nextInt();

        sc.close();

        System.out.println(Pow(x,n) % m);

    }

}



**8. Вычислите наименьшее общее кратное всех целых чисел от 1 до n.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Одно натуральное число **n** (**1** ≤ **n** ≤ **30**). | Выведите наименьшее общее кратное всех целых чисел от **1** до **n**. |
| 3 | 6 |

import java.util.Scanner;

public class Lab2\_8 {

    public static long nod(long a, long b){

        return (b == 0) ? a : nod(b, a % b);

    }

    public static long nok(long a, long b){

        return a \* b / nod(a,b);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        sc.close();

        long result = 1;

        for(int i = 2; i < n; ++i){

            result = nok(result,i);

        }

        System.out.println(result);

    }

}

