PROBLEM SET #1

Разрешено при решении использовать:

- примитивные типы данных,
- арифметические и логические операции,
- циклы (while, for, for each),
- условия (if, switch, тернарный оператор),
- массивы.

Нельзя использовать:

- String и все похожее на нее
- коллекции
- классы и объекты.

Можно использовать Arrays.toString для вывода массива на экран.

Ввод разрешено делать любым способом (либо args, либо Scanner). Вывод – только через System.out.print/println. При вводе разрешено использовать Integer.parseInt. Можно вместо ввода данных использовать Random для генерации случайных чисел.

Организация файлов

В репозитории создать папку problem-set-1. Каждый файл называть Task**XX**PS**N**, где XX - двузначный номер задачи (если номер задачи 1, то ее номер - 01), а N - номер problem-set.

Каждую задачу оформлять в виде отдельного класса. Можно в процессе еще решения использовать методы (static), но разумно. Пакеты в этом problem set не нужно использовать.

Внутри файла самыми первым строчкам в комментариях код должен быть вам подписан по образцу.

```
/**
* @author Mikhail Abramskiy
* 09-53a
* Problem Set 1 Task 01 (для вспомогательного класса указывайте для чего используется, например for Problem Set 1 Task 01)
*/
```

Дедлайн сдачи Problem Set 1 – 25 ноября 2017 года, 10:00.

Количество баллов за Problem Set 1: 9 (девять).

Просроченный дедлайн не отменяет сдачи задачи.

- Задачи, загруженные на неделю позже дедлайна, оцениваются в половину своей оценки. Еще позже – в 1/5 своей оценки.
- Задачи, загруженные вовремя, содержащие небольшие недочеты, разрешено будет исправить без потери балла.
- Задачи, загруженные вовремя, содержащие крупные нарушения, игнорирование запретов, необходимо будет исправить, теряя при этом половину оценки.
- За плагиат хотя бы одной задачи оценка обоих участников процесса будет равна ***30 сотым*** балла при необходимости все равно загрузить все задачи.

ЗАДАЧИ

01
Для введенного n вывести «трифорс» (пример ниже), n - высота каждого треугольника. Например, для n = 3

```
    *
   ***
  *****

  *       *
 ***     ***
***** *****
```

02
Вводятся целые k, m. Вывести целые числа между k и m, которые делятся на 3.

03
Подчитать квадратный корень числа x с точностью до 6 знака после запятой. Не использовать Math.

04
Разложить введенное число n на множители, вывести их на экран. Выводить все делители, не только уникальные (т.е., например, для 12 нужно вывести 2 2 3).

05
Для введенного целого n подсчитать

$$S = \sum_{m=1}^{n} \frac{((m-1)!)^2}{(2m)!}$$

06

Для введенного n приблизительно нарисовать символами круг радиуса n. Например, для n = 10:

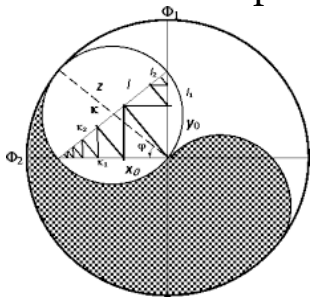
```
**********0**********
******0000000000******
****00000000000000****
***000000000000000***
**0000000000000000000**
**0000000000000000000**
*00000000000000000000*
*00000000000000000000*
*00000000000000000000*
*00000000000000000000*
00000000000000000000000
*00000000000000000000*
*00000000000000000000*
*00000000000000000000*
*00000000000000000000*
**0000000000000000000**
**0000000000000000000**
***000000000000000***
****00000000000000****
******0000000000******
**********0**********
```

(выглядит как овал, но только потому, что расстояние между буквами меньше, чем расстояние между строками, а если такое нарисовать, будет именно кружок)

07

Вывести на экран составленный из символов «Инь-Янь» (без маленьких кружочков). Размер задается с клавиатуры.

Картинка в помощь, она объясняет, из чего фактически состоит этот знак, на мелкие символы вы можете не обращать внимания.



08

Вычислить ряд с точностью 1e-9, |x| < 1.

$$\sum_{k=0}^{\infty} \frac{(-1)^k x^{4k+1}}{(2k)!(4k+1)}$$

09

Вычислить ряд с точностью 1e-9, |x| < 1.

$$\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{k!(2k+1)}$$

10
Локальным максимумом в последовательности назовем элемент, который больше предыдущего и следующего (если они есть). Проверить, что во входной последовательности существует четный по значению локальный максимум.

11
Проверить, что во входной последовательности целых чисел существует ровно два четных по значению локальных максимума.

12
Вводится число n. Построить число m, которое будет содержать только нечетные цифры числа n (в сохраненном порядке). Например, для n = 123456, m = 135.

13
Найдите сумму 1+11+111+1111+...1111...1, если последнее слагаемое состоит из n цифр

14
Вводится целое 2 < k < 9, затем вводится целое число n, которое можно интерпретировать как число в k-ичной системе счисления. Сконвертировать n в десятичную систему счисления.

15
Вводится n чисел. Проверить, что среди них существует ровно два таких числа, что длина (количество цифр) каждого из них равна 3 или 5, а их цифры либо все четные, либо все нечетные.

16
Вводится n чисел. Проверить, что существует ровно три числа, в котором цифры идут по убыванию.

17
Развернуть введенный массив (физически развернуть, а не «вывести в обратном порядке)

18
Вводится массив, затем число k, k < длина массива.
Поменять местами блоки в массиве от 0 до k-1 и от k до n-1
Пример:
Вход:
1 2 3 4 5 6 7 8
k = 4
Выход:
5 6 7 8 1 2 3 4

19
Вводится целочисленная квадратная матрица размера 2n+1. Заменить нулями элементы, расположенные на верхнем и нижнем треугольниках, образованных пересечением главной и побочной диагоналей. Правый и левый треугольники не трогать. Пример:

Вход:
1 2 3 4 5
2 3 4 1 2
0 3 2 3 1
9 2 3 1 4
3 8 0 8 6
Выход:
1 0 0 0 5
2 3 0 1 2
0 3 2 3 1
9 2 0 1 4
3 0 0 0 6

20 (считается за две)
Вводится квадратная матрица размера n. Привести ее к треугольному виду и вывести на экран. Не забудьте, что надо обрабатывать случай поиска ненулевого элемента в столбце (поиск – существование – квантор).

22
Подсчитать в двумерном массиве максимум по суммам элементов на главных диагоналях.
Пример
1 2 3
1 2 3
0 7 2
Главные диагонали: 1 2 2, 2 3 0, 3 1 7.
Суммы: 5, 5, 11. Максимум 11

23
Реализовать с помощью двумерных массивов умножение матриц.

24
Проверить, что в трехмерном массиве в каждом его двумерном массиве существует такая строка, что в ней все элементы делятся на три.