/\*\*

\* Gets a command-line argument (int), and prints all the divisors of the given number.

\*/

public class Divisors {

public static void main (String[] args) {

int userInt = Integer.parseInt(args[0]);

if (userInt > 0) {

for(int i = 1; i <= userInt; i++) {

if (userInt % i == 0) {

System.out.println(i); }

}

} else if (userInt < 0) {

for(int i = -1; i >= userInt; i--) {

if (userInt % i == 0) {

System.out.println(i);

}

}

} else {

System.out.println(userInt);

}

}

}

/\*\*

\* Prints a given string, backward. Then prints the middle character in the string.

\* The program expects to get one command-line argument: A string.

\*/

public class Reverse {

public static void main (String[] args){

String userInput = args[0];

for (int i = userInput.length() - 1; i >= 0; i--) {

System.out.print(userInput.charAt(i));

}

boolean isInputEven = userInput.length() % 2 == 0;

char mid = isInputEven ?

userInput.charAt((int) ((userInput.length() / 2) - 0.5)) : userInput.charAt(userInput.length() / 2);

System.out.println("\nThe middle character is " + mid);

}

}

/\*\*

\* Generates and prints random integers in the range [0,10),

\* as long as they form a non-decreasing sequence.

\*/

public class InOrder {

public static void main (String[] args) {

int num = -1;

int random = (int) (Math.random() \* 10);

do {

System.out.print(random + " ");

num = random;

random = (int) (Math.random() \* 10);

} while (random >= num);

}

}

/\*\*

\* Gets a command-line argument n (int), and prints an n-by-n damka board.

\*/

public class DamkaBoard {

public static void main(String[] args) {

int boardSize = Integer.parseInt(args[0]);

for (int i = 0; i < boardSize; i++) {

for (int j = 0; j < boardSize; j++) {

if (i % 2 == 0) {

System.out.print("\* ");

} else {

System.out.print(" \*");

}

}

System.out.println("");

}

}

}

/\* Gets a command-line argument (int), and checks if the given number is perfect.

\*/

public class Perfect {

public static void main (String[] args) {

int userInput = Integer.parseInt(args[0]);

int sum = 0;

boolean isPerfect = false;

String answer = " = ";

String numStr = "";

for (int i = 1; i < userInput; i++) {

if (userInput % i == 0) {

sum += i;

// Convert the divisor to str and concatinate

numStr = Integer.toString(i);

answer = (answer + numStr + " + ");

}

}

// Remove the " +" of the end

answer = answer.substring(0, answer.length() - 3);

if (sum == userInput) {

isPerfect = true;

System.out.println(userInput + " is a perfect number since " + userInput +

answer);

} else {

System.out.println(userInput + " is not a perfect number");

}

}

}

import java.util.Random;

/\*\*

\* Computes some statistics about families in which the parents decide

\* to have children until they have at least one child of each gender.

\* The program expects to get two command-line arguments: an int value

\* that determines how many families to simulate, and an int value

\* that serves as the seed of the random numbers generated by the program.

\* Example usage: % java OneOfEachStats 1000 1

\*/

public class OneOfEachStats {

public static void main (String[] args) {

// Gets the two command-line arguments

int T = Integer.parseInt(args[0]);

int seed = Integer.parseInt(args[1]);

// Initailizes a random numbers generator with the given seed value

Random generator = new Random(seed);

//// In the previous version of this program, you used a statement like:

//// double rnd = Math.random();

//// Where "rnd" is the variable that stores the generated random value.

//// In this version of the program, replace this statement with:

//// double rnd = generator.nextDouble();

//// This statement will generate a random value in the range [0,1),

//// just like you had in the previous version, except that the

//// randomization will be based on the given seed.

//// This is the only change that you have to do in the program.

// User inputs T - number of families to form (amount of runs)

int countChildrenPerRun = 1;

boolean isOneOfEach = false; // Tool to break while

char firstChild = ' ';

char child = firstChild;

int totalCount = 0;

int twoChilds = 0, threeChilds = 0, fourOrMoreChilds = 0; // Counts the families

// Each "for" creates one scenario

for (int t = 0; t < T; t++) {

// The "while" counts how many children until at least

// one of each gender.

firstChild = generator.nextDouble() > 0.5 ? 'b' : 'g';

while (!isOneOfEach) {

child = generator.nextDouble() > 0.5 ? 'b' : 'g';

if (child != firstChild) {

isOneOfEach = true;

}

countChildrenPerRun += 1;

}

// Count families

if (countChildrenPerRun == 2) twoChilds++;

else if (countChildrenPerRun == 3) threeChilds++;

else if (countChildrenPerRun > 3) fourOrMoreChilds++;

totalCount += countChildrenPerRun;

// Reset before next run

isOneOfEach = false;

countChildrenPerRun = 1;

}

double average = (double) (totalCount) / T;

String mostCommon = "";

if (twoChilds >= threeChilds && twoChilds >= fourOrMoreChilds) {

mostCommon = "2";

} else if (threeChilds >= twoChilds && threeChilds >= fourOrMoreChilds) {

mostCommon = "3";

} else {

mostCommon = "4 or more";

}

System.out.println("Average: " + average +

" children to get at least one of each gender.");

System.out.println("Number of families with 2 children: " + twoChilds);

System.out.println("Number of families with 3 children: " + threeChilds);

System.out.println("Number of families with 4 or more children: " +

fourOrMoreChilds);

System.out.println("The most common number of children is " +

mostCommon + ".");

}

}