/\*\*

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

\*/

public class Divisors {

public static void main (String[] args) {

// checking if args is greater than zero

if (args.length > 0) {

// set args value into an int

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

// runs from 1 to the given number and prints every

// number that is a divisor of the given number

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

if (num % i == 0) {

System.out.println(i);

}

}

}

}

}

/\*\*

\* 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){

// checking if args is greater than zero

if (args.length > 0) {

// put the args into a string

String str = args[0];

// runs from the last index of the string

// to the first index

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

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

}

// calculating the middle index

int middle = 0;

if (str.length() % 2 == 0) {

middle = ( str.length() / 2 ) - 1;

}

else {

middle = ( str.length() / 2 );

}

//printing the character at the middle index

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

}

}

}

/\*\*

\* 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) {

// creates a random number

int N = 10, firstNum = 0, secondNum = 0;

double r = Math.random();

firstNum = (int) (r \* N);

secondNum = firstNum;

//running till secondNum is smaller than firstNum

do {

firstNum = secondNum;

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

r = Math.random();

secondNum = (int) (r \* N);

}

while (secondNum >= firstNum);

System.out.println();

}

}

/\*\*

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

\*/

public class Perfect {

public static void main (String[] args) {

// checking if args is greater than zero

if (args.length > 0)

{

// putting the given number in an

// integer N and a sun

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

int sum = 0;

//creating a String that holds

//all the divisors

String divisors = "";

//running through all the numbers form

//1 to N - 1

for (int i = 1; i < N - 1; i++) {

//checking if N % i == 0

if (N % i == 0) {

//adding i to sum and adding to the

//divisors String

sum += i;

divisors += i + " + ";

}

}

//checks if N is equal to sum

//and print accordingly

if (N == sum) {

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

" = " + divisors.substring(0, divisors.length() - 3));

}

else {

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

}

}

}

}

/\*\*

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

\*/

public class DamkaBoard {

public static void main(String[] args) {

// checking if args is greater than zero

if (args.length > 0)

{

// putting the given number n

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

//creating a bool arg that holds

//if the first char is asterisk

//or not

boolean isAsterisk = true;

//running n times on n times

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

//line start with an asterisk if

//the line is even

isAsterisk = i % 2 == 0;

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

if (isAsterisk) {

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

}

else {

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

}

}

//after we finish the line we

//need to get down to a new line

System.out.println();

}

}

}

}

/\*\*

\* Simulates the formation of a family in which the parents decide

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

\* If the probability is 0 it is a boy, otherwise it is a girl

\*/

public class OneOfEach {

public static void main (String[] args) {

// creates a random number

int N = 2, probability;

double r = 0;

//creating a count

int count = 0;

//creating 2 bool args that holds

//whether a boy or a girl were born

boolean isBoy = false, isGirl = false;

//runnning untill there are a boy

//and a girl

while (!isBoy || !isGirl) {

//adding 1 to the count

count++;

//calculating the probability

r = Math.random();

probability = (int) (r \* N);

//checking if it is a girl or a boy

if (probability == 0) {

System.out.print("b ");

isBoy = true;

}

else {

System.out.print("g ");

isGirl = true;

}

}

//printing how many children there are

System.out.println("\nYou made it... and you now have " + count + " children.");

}

}

/\*\*

\* 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 one command-line argument: an int value

\* that determines how many families to simulate.

\*/

public class OneOfEachStats1 {

public static void main (String[] args) {

// checking if args is greater than zero

if (args.length > 0) {

// set args value into an int

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

// creates a random number

int N = 2, probability;

double r = 0;

//creating a count and a avg

int count = 0;

double avg = 0.0;

//creating args of number of families with

//2, 3, 4 or more children and mode

int twoChildrenCount = 0, threeChildrenCount = 0, fourOrMoreChildrenCount = 0;

int max = 0;

String mode = "";

//creating 2 bool args that holds

//whether a boy or a girl were born

boolean isBoy = false, isGirl = false;

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

//setting values in all the args

isBoy = false;

isGirl = false;

count = 0;

//runnning untill there are a boy

//and a girl

while (!isBoy || !isGirl) {

//adding 1 to the count

count++;

//calculating the probability

r = Math.random();

probability = (int) (r \* N);

//checking if it is a girl or a boy

if (probability == 0) {

isBoy = true;

}

else {

isGirl = true;

}

}

//checking how many children are in the familiy

switch (count) {

case 2:

twoChildrenCount++;

break;

case 3:

threeChildrenCount++;

break;

default:

fourOrMoreChildrenCount++;

break;

}

//adding the count to avg

avg += (double) count;

}

//calculating the max number of families

max = Math.max(twoChildrenCount, threeChildrenCount);

max = Math.max(max, fourOrMoreChildrenCount);

//setting the correct string of mode

if (max == twoChildrenCount) {

mode = "2.";

}

else if (max == threeChildrenCount) {

mode = "3.";

}

else {

mode = "4 or more.";

}

//calculating the avg

avg /= (double) T;

//printing evertything

System.out.println("Average: " + avg + " children to get at least one of each gender.");

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

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

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

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

}

}

}

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) {

// checking if args is greater than zero

if (args.length > 0) {

// 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);

// creates a random number

int N = 2, probability;

double r = 0;

//creating a count and a avg

int count = 0;

double avg = 0.0;

//creating args of number of families with

//2, 3, 4 or more children and mode

int twoChildrenCount = 0, threeChildrenCount = 0, fourOrMoreChildrenCount = 0;

int max = 0;

String mode = "";

//creating 2 bool args that holds

//whether a boy or a girl were born

boolean isBoy = false, isGirl = false;

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

//setting values in all the args

isBoy = false;

isGirl = false;

count = 0;

//runnning untill there are a boy

//and a girl

while (!isBoy || !isGirl) {

//adding 1 to the count

count++;

//calculating the probability

r = generator.nextDouble();

probability = (int) (r \* N);

//checking if it is a girl or a boy

if (probability == 0) {

isBoy = true;

}

else {

isGirl = true;

}

}

//checking how many children are in the familiy

switch (count) {

case 2:

twoChildrenCount++;

break;

case 3:

threeChildrenCount++;

break;

default:

fourOrMoreChildrenCount++;

break;

}

//adding the count to avg

avg += (double) count;

}

//calculating the max number of families

max = Math.max(twoChildrenCount, threeChildrenCount);

max = Math.max(max, fourOrMoreChildrenCount);

//setting the correct string of mode

if (max == twoChildrenCount) {

mode = "2.";

}

else if (max == threeChildrenCount) {

mode = "3.";

}

else {

mode = "4 or more.";

}

//calculating the avg

avg /= (double) T;

//printing evertything

System.out.println("Average: " + avg + " children to get at least one of each gender.");

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

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

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

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

}

}

}