01. Smallest of Three Numbers

package Methods;

import java.util.Arrays;

import java.util.Scanner;

public class Ex1SmallestOfThreeNumbers {

public static void smallestNumber(int num1, int num2, int num3) {

int[] array = {num1, num2, num3};

Arrays.sort(array);

System.out.println(array[0]);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

smallestNumber(Integer.parseInt(scanner.nextLine()), Integer.parseInt(scanner.nextLine()), Integer.parseInt(scanner.nextLine()));

}

}

## 02. Vowels Count

package Methods;

import java.util.Scanner;

public class Ex2VowelCount {

public static int vowelCount(String inp, char[] input) {

int counter = 0;

input = inp.toLowerCase().toCharArray();

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

if (input[i] == 97 || input[i] == 101 || input[i] == 105

|| input[i] == 111 || input[i] == 113 || input[i] == 117 || input[i] == 121) {

counter++;

}

}

return counter;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String inp = scanner.nextLine();

char[]input=inp.toLowerCase().toCharArray();

System.out.println(vowelCount(inp,input));

}

}

## 03. Characters in Range

package Methods;

import java.util.Scanner;

public class Ex3CharactersInRange {

public static void charactersInRange(char first, char second) {

if (first > second) {

for (int i = second+1; i < first; i++) {

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

}

}

for (int i = first + 1; i < second; i++) {

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

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

char first = scanner.next().charAt(0);

char second = scanner.next().charAt(0);

charactersInRange(first, second);

}

}

## 04. Password Validator

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex4PasswordValidator {  
  
 **public static void** passwordValidatorFalse(String input) {  
 **boolean** between6And10 = **true**;  
 **boolean** onlyLettersAndDigits = **true**;  
 **boolean** haveAtLeast2digits = **true**;  
 **int** count = 0;  
 **if** (input.length() >= 6 && input.length() <= 10) {  
  
 } **else** {  
 between6And10 = **false**;  
 }  
  
 **for** (**int** i = 0; i < input.length(); i++) {  
 **if** ((input.charAt(i) >= 65 && input.charAt(i) <= 90) || (input.charAt(i) >= 97 && input.charAt(i) <= 122) || (input.charAt(i) >= 48 && input.charAt(i) <= 57)) {  
  
 } **else** {  
 onlyLettersAndDigits = **false**;  
 **break**;  
 }  
  
 }  
 **for** (**int** i = 0; i < input.length(); i++) {  
  
 **if** (input.charAt(i) >= 48 && input.charAt(i) <= 57) {  
 count++;  
  
 }  
 }  
 **if** (count < 2) {  
 haveAtLeast2digits = **false**;  
 }  
  
 **if** (between6And10 && onlyLettersAndDigits && haveAtLeast2digits) {  
 System.***out***.println(**"Password is valid"**);  
 }  
 **if** (!between6And10) {  
 System.***out***.println(**"Password must be between 6 and 10 characters"**);  
 }  
 **if** (!onlyLettersAndDigits) {  
 System.***out***.println(**"Password must consist only of letters and digits"**);  
 }  
 **if** (!haveAtLeast2digits) {  
 System.***out***.println(**"Password must have at least 2 digits"**);  
 }  
 }  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 String input = scanner.nextLine();  
 *passwordValidatorFalse*(input);  
 }  
}

## 05. Add and Subtract

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex5AddAndSubtract {  
 **public static int** Add(**int** num1,**int** num2){  
 **int** sum = num1+num2;  
 **return** sum;  
 }  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 **int** num1 = Integer.*parseInt*(scanner.nextLine());  
 **int** num2 = Integer.*parseInt*(scanner.nextLine());  
 **int** num3 = Integer.*parseInt*(scanner.nextLine());  
 **int** result = *Add*(num1,num2)-num3;  
 System.***out***.println(result);  
 }  
}

## 06. Middle Characters

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex6MiddleCharacters {  
 **public static void** MiddleChar(String input) {  
 **if** (input.length() % 2 == 0) {  
 **for** (**int** i = input.length() / 2; i <= (input.length() / 2) + 1; i++) {  
 System.***out***.print(input.charAt(i - 1));  
 }  
 } **else** {  
 **for** (**int** i = input.length() / 2; i < (input.length() / 2) + 1; i++) {  
 System.***out***.print(input.charAt(i));  
 }  
  
 }  
 }  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 String input = scanner.nextLine();  
  
 *MiddleChar*(input);  
 }  
}

## 07. NxN Matrix

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex7NxNMatrix {  
 **public static void** NxNmatrix(**int** n) {  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** j = 0; j < n; j++) {  
 System.***out***.print(n + **" "**);  
 }  
 System.***out***.println();  
  
 }  
 }  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 **int** n = Integer.*parseInt*(scanner.nextLine());  
 *NxNmatrix*(n);  
  
 }  
}

## 08. Factorial Division

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex8FactorialDivision {  
 **public static long** factorial(**long** num) {  
 **long** fact = 1;  
 **for** (**long** i = 1; i <= num; i++) {  
 fact = fact \* i;  
  
 }  
 **return** fact;  
 }  
  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 **long** num1 = Integer.*parseInt*(scanner.nextLine());  
 **long** num2 = Integer.*parseInt*(scanner.nextLine());  
 **long** fact1 = *factorial*(num1);  
 **long** fact2 = *factorial*(num2);  
 **double** result = *factorial*(num1)\*1.0 / *factorial*(num2);  
 System.***out***.printf(**"%.2f"**, result);  
  
  
 }  
}

## 09. Palindrome Integers

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex9PalindromeIntegers {  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 String input = scanner.nextLine();  
 **int** num = Integer.*parseInt*(input), reversedInteger = 0, remainder, originalInteger;  
  
  
 *// reversed integer is stored in variable* **while** (!input.equals(**"END"**)) {  
 num = Integer.*parseInt*(input);  
 originalInteger = num;  
 **while** (num != 0) {  
 remainder = num % 10;  
 reversedInteger = reversedInteger \* 10 + remainder;  
 num /= 10;  
  
 }  
  
 *// palindrome if orignalInteger and reversedInteger are equal* **if** (originalInteger == reversedInteger) {  
 System.***out***.println(**"true"**);  
 } **else** {  
 System.***out***.println(**"false"**);  
 }  
 reversedInteger = 0;  
 input = scanner.nextLine();  
  
 }  
 }  
}

## 10. Top Number

**package** Methods;  
  
**import** java.util.Scanner;  
  
**public class** Ex10TopIntegers {  
 **public static void** topNumbers(**int** n) {  
 **for** (**int** i = 1; i <= n; i++) {  
 **int** num = i;  
 **int** sum = 0;  
 **boolean** isOdd = **false**;  
 **while** (num > 0) {  
 **if** ((num%10)%2!=0){  
 isOdd = **true**;  
 }  
 sum = sum + num % 10;  
 num = num / 10;  
  
 }  
 **if** (sum%8==0&&isOdd){  
 System.***out***.println(i);  
 }  
 }  
  
 }  
  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 **int** n = Integer.*parseInt*(scanner.nextLine());  
 *topNumbers*(n);  
 }  
}

## 11. Array Manipulator

**package** Methods;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.util.Arrays;  
  
**public class** Ex11ArrayManipul {  
 **public static void** main(String[] args) **throws** IOException {  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 **int**[] number = Arrays.*stream*(reader.readLine().split(**" "**)).mapToInt(Integer::*parseInt*).toArray();  
  
 String command = reader.readLine();  
  
 **while** (!command.equals(**"end"**)) {  
  
 String[] cmdArgs = command.split(**" "**);  
  
 **if** (cmdArgs[0].equals(**"exchange"**)) {  
 **int** index = Integer.*parseInt*(cmdArgs[1]);  
 **if** (index >= 0 && index < number.**length**) {  
 *exchange*(number, index);  
 } **else** {  
 System.***out***.println(**"Invalid index"**);  
 }  
 } **else if** (cmdArgs[0].equals(**"max"**)) {  
 **if** (cmdArgs[1].equals(**"even"**)) {  
 **int** index = *findEvenIndex*(number);  
 **if** (index != -1) {  
 System.***out***.println(index);  
 } **else** {  
 System.***out***.println(**"No matches"**);  
  
 }  
 } **else** {  
 **int** index = *findOddIndex*(number);  
 **if** (index != -1) {  
 System.***out***.println(index);  
 } **else** {  
 System.***out***.println(**"No matches"**);  
  
 }  
  
 }  
 } **else if** (cmdArgs[0].equals(**"min"**)) {  
 **if** (cmdArgs[1].equals(**"even"**)) {  
 **int** index = *findMinEvenIndex*(number);  
 **if** (index != -1) {  
 System.***out***.println(index);  
 } **else** {  
 System.***out***.println(**"No matches"**);  
  
 }  
 } **else** {  
 **int** index = *findMinOddIndex*(number);  
 **if** (index != -1) {  
 System.***out***.println(index);  
 } **else** {  
 System.***out***.println(**"No matches"**);  
  
 }  
  
 }  
 } **else if** (cmdArgs[0].equals(**"first"**)) {  
 **int** count = Integer.*parseInt*(cmdArgs[1]);  
 **if** (cmdArgs[2].equals(**"even"**)) {  
 **if** (count <= number.**length**) {  
 *printFirstEven*(number, count);  
 } **else** {  
 System.***out***.println(**"Invalid count"**);  
 }  
 } **else** {  
 **int** oddCount = Integer.*parseInt*(cmdArgs[1]);  
 **if** (oddCount <= number.**length**) {  
 *printFirstOdd*(number, count);  
 } **else** {  
 System.***out***.println(**"Invalid count"**);  
 }  
  
  
 }  
 } **else if** (cmdArgs[0].equals(**"last"**)) {  
 **int** count = Integer.*parseInt*(cmdArgs[1]);  
 **if** (cmdArgs[2].equals(**"even"**)) {  
 **if** (count <= number.**length**) {  
 *printLastEven*(number, count);  
 } **else** {  
 System.***out***.println(**"Invalid count"**);  
 }  
 } **else** {  
 **int** oddCount = Integer.*parseInt*(cmdArgs[1]);  
 **if** (oddCount <= number.**length**) {  
 *printLastOdd*(number, count);  
 } **else** {  
 System.***out***.println(**"Invalid count"**);  
 }  
  
  
 }  
 }  
  
 command = reader.readLine();  
  
 }  
 *printArray*(number);  
 }  
  
 **private static void** printLastEven(**int**[] number, **int** count) {  
 **int**[] arr = **new int**[number.**length**];  
 **for** (**int** i = number.**length** - 1; i >= 0; i--) {  
 **if** (number[i] % 2 == 0 && count > 0) {  
 count--;  
 arr[i] = number[i];  
 } **else** {  
 arr[i] = -1;  
 }  
  
 }  
 *printArray*(arr);  
  
  
 }  
 **private static void** printLastOdd(**int**[] number, **int** count) {  
 **int**[] arr = **new int**[number.**length**];  
 **for** (**int** i = number.**length** - 1; i >= 0; i--) {  
 **if** (number[i] % 2 != 0 && count > 0) {  
 count--;  
 arr[i] = number[i];  
 } **else** {  
 arr[i] = -1;  
 }  
  
 }  
 *printArray*(arr);  
  
  
 }  
  
 **private static void** printFirstOdd(**int**[] number, **int** count) {  
 **int**[] arr = **new int**[number.**length**];  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (number[i] % 2 != 0 && count > 0) {  
 count--;  
 arr[i] = number[i];  
 } **else** {  
 arr[i] = -1;  
 }  
  
 }  
 *printArray*(arr);  
  
  
 }  
  
 **private static void** printFirstEven(**int**[] number, **int** count) {  
 **int**[] arr = **new int**[number.**length**];  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (number[i] % 2 == 0 && count > 0) {  
 count--;  
 arr[i] = number[i];  
 } **else** {  
 arr[i] = -1;  
 }  
  
 }  
 *printArray*(arr);  
 }  
  
 **private static int** findMinOddIndex(**int**[] number) {  
 **int** index = -1;  
  
 **int** maxNum = Integer.***MAX\_VALUE***;  
  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (maxNum >= number[i] && number[i] % 2 != 0) {  
 maxNum = number[i];  
 index = i;  
  
 }  
  
 }  
  
 **return** index;  
 }  
  
 **private static int** findMinEvenIndex(**int**[] number) {  
 **int** index = -1;  
  
 **int** maxNum = Integer.***MAX\_VALUE***;  
  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (maxNum >= number[i] && number[i] % 2 == 0) {  
 maxNum = number[i];  
 index = i;  
  
 }  
  
 }  
  
 **return** index;  
 }  
  
 **private static int** findOddIndex(**int**[] number) {  
 **int** index = -1;  
  
 **int** maxNum = Integer.***MIN\_VALUE***;  
  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (maxNum <= number[i] && number[i] % 2 != 0) {  
 maxNum = number[i];  
 index = i;  
  
 }  
  
 }  
  
 **return** index;  
  
 }  
  
 **private static int** findEvenIndex(**int**[] number) {  
 **int** index = -1;  
  
 **int** maxNum = Integer.***MIN\_VALUE***;  
  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (maxNum <= number[i] && number[i] % 2 == 0) {  
 maxNum = number[i];  
 index = i;  
  
 }  
  
 }  
  
 **return** index;  
 }  
  
 **private static void** printArray(**int**[] number) {  
 System.***out***.print(**"["**);  
 **boolean** printFirst = **true**;  
 **for** (**int** i = 0; i < number.**length**; i++) {  
 **if** (number[i] != -1) {  
 **if** (printFirst) {  
  
 System.***out***.print(number[i]);  
 printFirst = **false**;  
 } **else** {  
 System.***out***.print(**", "** + number[i]);  
 }  
 }  
 }  
 System.***out***.println(**"]"**);  
 }  
  
 **private static void** exchange(**int**[] number, **int** index) {  
  
 **int**[] firstSide = **new int**[index + 1];  
 **int**[] secondSide = **new int**[number.**length** - (index + 1)];  
  
 **for** (**int** i = 0; i <= index; i++) {  
 firstSide[i] = number[i];  
  
 }  
  
 **for** (**int** i = index + 1; i < number.**length**; i++) {  
 secondSide[i - (index + 1)] = number[i];  
  
 }  
  
 **for** (**int** i = 0; i < secondSide.**length**; i++) {  
 number[i] = secondSide[i];  
  
 }  
  
 **for** (**int** i = 0; i < firstSide.**length**; i++) {  
 number[i + secondSide.**length**] = firstSide[i];  
  
 }  
 }  
}