e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Loops-Exercises

1. Numbers from 0 to 100

Write a function that prints the numbers from 1 to 100, each on a new line.

2. Numbers N to 0 in reverse order

Write a function that accepts a positive number n and prints the numbers from n to 1 in reverse order (from largest to smallest).

Sample input:

input	output
10	10 9 8 7 6 5 4 3 2 1 0
5	5 4 3 2 1

	0

Guidelines

- Do one for loop from n to 0, but instead of increasing the variable i by 1, decrease it by 1 at each iteration of the loop.

3. Numbers 1 to N through 2

Write a program that accepts a number n and prints the numbers from 1 to n through 2 (with step 2).

Sample input:

input	output
10	1 3 5 7 9
5	1 3 5

Guidelines

- Do for loop 1 to n (inclusive) and set step 2. This means that at each iteration of the loop, variable i will increase its value by 2 instead of 1.
- In the body of the loop, print the variable i.

4. Numbers 1 to N through M

Write a function that takes two numbers n and m and prints the numbers from 1 to n through m (with step m).

Sample input:

input	output
10 **2**	1 3 5 7 9
8 **3**	1 4 7

5. Letters in a word

Write a function that accepts text and prints each letter on a new line.

Sample input:

input	output
hello	h e l l o
Bulgaria	B u l g a r i a

6. Sum of vowels

Write a function that accepts text (string) and calculates and prints the sum of the values of the vowel letters according to the table below:

letter	a	e	i	o	u
value	1	2	3	4	5

Sample input:

input	output
hello	6
hi	3
bye	2
zzzz	0

Guidelines

- Create a variable for the sum of letters.
- Do for loop from 0 to word.length (the length of the text).
- Check that each letter word[i] is a vowel and add its value to the sum accordingly.

7. Clock

Write a function that prints the hours of the day from 0:0 to 23:59, each on a separate line. Hours must be written in the format '{hour}:{minutes}'.

Sample input:

input	output
	0:0 0:1

	0:2 ... 23:58 23:59

8. Multiplication table

Print the multiplication table for numbers 1 through 10 in the following format:

"{first multiplier} * {second multiplier} = {result}".

Sample input:

input	output
	1 * 1 = 1 1 * 2 = 2 1 * 3 = 3 ... 10 * 8 = 80 10 * 9 = 90 10 * 10 = 100

9. Sum of Even Numbers

Write a program that receives an integer 'n' and calculates the sum of the first 'n' even numbers. Display the result on the console.

input	output
3	12
5	30
1	2
0	0
10	10

10. Factorial Calculation

Write a program that receives a positive integer 'n' and calculates its factorial.

input	output
5	120
3	6
0	1
1	1
10	3628800

11. Number Reversal

Write a program that receives an integer 'n' and prints its reverse.

input	output
123	321
9876	6789
505	505
10203	30201
7	7

12. Fibonacci Sequence Sum

Write a program that receives a positive integer 'n' and calculates the sum of the first 'n' items in the Fibonacci sequence.

input	output
3	4
5	12
1	1
0	0
10	143

13. Palindrome Check

Write a program that receives a string from the console and checks whether it is a palindrome (reads the same forwards and backwards) or not.

input	output
radar	true
hello	false
racecar	true
java	false
madam	true

14. Armstrong Number Check

Write a program that receives an integer 'n' and checks whether it is an Armstrong number or not. An Armstrong number is a number that is equal to the sum of its own digits raised to the power of the count of digits.

Input: An integer 'n' to be checked for being an Armstrong number.

Output: "Armstrong" if 'n' is an Armstrong number, "Not Armstrong" otherwise.

input	output
153	true
370	true
123	false
407	true
1634	true

15. Collatz Conjecture

Write a program that receives a positive integer 'n' and calculates and prints the Collatz sequence for 'n'. The Collatz sequence is defined as follows:

- If 'n' is even, divide it by 2.
- If 'n' is odd, multiply it by 3 and add 1.
- Repeat the process until 'n' becomes 1.

input	output
6	6 3 10 5 16 8 4 2 1
12	12 6 3 19 5 16 8 4 2 1
9	9 28 14 7 22 11 34 17 52 26 13 49 29 19 5 16 8 4 2 1
200	200 100 50 25 76 ...
15	15 46 23 70 35 106 ...

16. Hollow Rectangle Pattern

Input: Two positive integers 'n' and 'm' representing the number of rows and columns.

input	output

4
6
	`******`
	`*    *`
	`*    *`
	`******`

2 **2**	`**` `**`

3
4
	`****`
	`*  *`
	`****`

17. New Building

Write a function that displays on the console a building (from top to bottom), while the following conditions are met:

- On each even floor there are only offices
- On each odd floor there are only apartments
- Each apartment is indicated as follows: A{floor number}{apartment number}, apartment numbers start from 0.
- Each office is indicated as follows: O{floor number}{office number}, office numbers also start at 0.

- On the top floor there are always apartments, and they are larger than the others, so in front of their number it says 'L' instead of 'A'. If there is only one floor, then there are only large apartments!

You accept 2 parameters - the number of floors and the number of rooms per floor.

Sample input:

input	output	explanation
6 **4**	L60 L61 L62 L63 A50 A51 A52 A53 O40 O41 O42 O43 A30 A31 A32 A33 O20 O21 O22 O23 A10 A11 A12 A13	We have a total of 6 floors, with 4 rooms per floor. Odd floors have only apartments and even-numbered floors only offices.
3 **3**	L30 L31 L32 O20 O21 O22 A10 A11 A12	
4 **4**	L40 L41 L42 L43 A30 A31 A32 A33 O20 O21 O22 O23 A10 A11 A12 A13	

18. Magic Number

Write a function that checks all possible combinations of a pair of numbers in the range of two given numbers. Print in which line is the combination whose sum of the numbers is equal to a given magic number. If there is no combination matching the condition print a message that magic number was not found.

Three parameters:

- First – beginning of the interval – integer in the range [1...999]
- Second – end of the interval – integer in the interval [greater than the first number... 1000]
- Third – the magic number – an integer in the range [1...10000]

Output

One line should be printed on the console, according to the result:

- If a combination whose sum of numbers is equal to the magic number is found
  - "Combination {sequence number} - ({first number} + {second number} = {magic number})"
- If no combination was found matching the condition
  - "{the number of combinations} combinations - neither equals {magic number}"

Sample input:

input	output	explanation
1 **10** **5**	Combination 4 - (1 + 4 = 5)	All combinations of two numbers between 1 and 10 are: 1 1, 1 2, 1 3, 1 4, 1 5, ... 2 1, 2 2, ... 4 9, 4 10, 5 1 ... 10 9, 10 10 The first combination whose sum of numbers is equal to the magic number 5 is the fourth (1 and 4)
23 **24** **20**	4 combinations - neither equals 20	All combinations of two numbers between 23 and 24 are: 23 23, 23 24, 24 23, 24 24 (total 4) There are no pairs of numbers whose sum is equal to the magic 20
1 **2** **3**	Combination 2 - (1 + 2 = 3)	

19. Padawan Equipment

Yoda is starting his newly created Jedi academy. So, he asked Master George Lucas to buy the needed equipment. The number of items depends on how many students will sign up. The equipment for the Padawan contains lightsabers, belts, and robes.

You will be given the amount of money George Lucas has, the number of students, and the prices of each item. You must help George Lucas calculate if the money he has is enough to buy all of the equipment or how much more money he needs.

Because the lightsabers sometimes break, George Lucas should buy 10% more, rounded up to the next integer. Also, every sixth belt is free.

The input data consist of exactly 5 lines:

- The amount of money George Lucas has – the floating-point number in the range [0.00…1,000.00].
- The count of students – integer in the range [0…100].
- The price of lightsabers for a single saber – the floating-point number in the range [0.00…100.00].
- The price of robes for a single robe – the floating-point number in the range [0.00…100.00].
- The price of belts for a single belt – the floating-point number in the range [0.00…100.00].

The output should be printed on the console.

- If the calculated price of the equipment is less or equal to the money George Lucas has:
  - "The money is enough - it would cost {the cost of the equipment}lv."
- If the calculated price of the equipment is more than the money George Lucas has:
  - "George Lucas will need {neededMoney}lv more."

All prices must be rounded to two digits after the decimal point.

input	output	explanation
100 **2** **1.0** **2.0** **3.0**	The money is enough - it would cost 13.00lv.	Needed equipment for 2 padawans: sabresPrice * (studentsCount + 10%) + robesPrice * (studentsCount) + beltsPrice * (studentsCount - freeBelts) 1*(3) + 2*(2) + 3*(2) = 13.00 13.00 <= 100 – the money will be enough.
100 **42** **12.0** **4.0** **3.0**	George Lucas will need 737.00lv more.	Needed equipment for 42 padawans: 12*47 + 4*42 + 3*35 = 837.00 837 > 100 – need 737.00 lv. more.

20. Rage Expenses

As a MOBA challenger player, Peter has the bad habit of trashing his PC when he loses a game and rage quits. His gaming setup consists of a headset, mouse, keyboard, and display. You will receive Peter's lost games count.

- Every second lost game, Peter trashes his headset.
- Every third lost game, Peter trashes his mouse.
- When Peter trashes both his mouse and headset in the same lost game, he also trashes his keyboard.
- Every second time he trashes his keyboard, he also trashes his display.

You will receive the price of each item in his gaming setup. Calculate his rage expenses for renewing his gaming equipment.

- On the first input line - lost games count – integer in the range [0, 1000].
- On the second line – headset price - the floating-point number in the range [0, 1000].
- On the third line – mouse price - the floating-point number in the range [0, 1000].
- On the fourth line – keyboard price - the floating-point number in the range [0, 1000].
- On the fifth line – display price - the floating-point number in the range [0, 1000].

As output, you must print Peter's total expenses: "Rage expenses: {expenses} lv."

input	output	explanation
7 **2** **3** **4** **5**	Rage expenses: 16.00 lv.	Trashed headset -> 3 times Trashed mouse -> 2 times Trashed keyboard -> 1 time Total: 6 + 6 + 4 = 16.00 lv;
23 **12.50**	Rage expenses: 608.00 lv.	

21.50 **40** **200**		

21. Refactor Sum of Odd Numbers

You are assigned to **find and fix the bugs** in an existing piece of code, using the **debugger**. You should trace the program execution to find the lines of code that produce incorrect or unexpected results.

You are given a program (existing source code) that prints the next **n odd numbers** (starting from 1) and on the **last row**, prints the **sum of them**.

Examples

Input	Output		Input	Output
5	1 3 5 7 9 Sum: 25		3	1 3 5 Sum: 9

SumOfOddNumbers.java

```
Scanner sc = new Scanner(System.in);
int n = Integer.parseInt(sc.nextLine());
int sum = 1;
for (int i = 0; i <= n; i++) {
   System.out.print(2 * i + 1);
   sum += 2 * i;
}
System.out.print"Sum: %d%n", sum);
```

22. Numbers up to 1000, ending in 7

Write a program that prints the numbers in the range **[1... 1000]**, which **end in 7**.

Input	Output
	7 17 27

	…
	997

23. Numbers up to 1000, ending in n

Write a program that prints the numbers in the range **[1… 1000]**, which **end in n**.

Input	Output	Input	Output
6	6	8	8
	16		18
	26		28
	…		…
	996		998

24. Encoding

Write a program that receive an integer. On the console should be printed as many lines as there are digits in the number. Each line is formed from the digits of the reversed number. A symbol must be printed on each line with the following conditions:

- the symbol to be printed is from the ASCII table. Its decimal ASCII code **is formed as 33 is added to the digit of the number entered that corresponds to a given line.**
- The symbol is printed **as many times as the digit**
- if for a given **line corresponds digit 0 on this line print "ZERO"**

Input	Output	Explanation
2049	*********	The number 2049 is four-digit so we will print 4 lines. In the first line corresponds the number 9. Add 33 to 9 and get 42. This is the decimal ASCII code of the character that we need to print on the first line. From the ASCII table we find that on 42 corresponds symbol *. Since the first row corresponds to the digit 9 we print * 9 times. The number 4 is for the second row. 4+33=37. From the ASCII table we find that the print symbol is %. We print % 4 times. On the third line we got 0. In this case we print a single ZERO. The last digit of the number is 2. 2+33=35. From the ASCII table we find the print symbol- # print it 2 times.
	%%%%	
	ZERO	
	##	

9347439	********* $$$ %%%% (((((((%%%% $$$ *********	

Tips:

1. To take the last digit of the number, use modulo by 10 (num %10) **and save it in one variable. Then** remove the last digit of the number by dividing by 10 (**num** / 10**) so that next time you can take the last digit again.**

25. Coins and Notes

We have banknotes and coins of **1lv.**, 2lv**.** and 5lv**.** Write a program that receives number of banknotes and coins and the target amount and displays all possible ways in which the amount can be paid with the available money.

The input contains **exactly 4 parameters**:

- **Number of** coins of **1lv. - positive integer;**
- **Number of** coins of **2lv. - positive integer;**
- **Number** of banknotes of **BGN 5 - positive integer;**
- **Sum - positive integer** in the range [**1… 1000**];

Print **all combinations of the given denominations forming the sum**, formatted as follows:

- **"{1 count} * 1 lv. + {2 count} * 2 lv. + {5 count} * 5 lv. = {sum} lv."**

Input	**Output**
3 **2** **3** **10**	0 * 1 lv. + 0 * 2 lv. + 2 * 5 lv. = 10 lv. 1 * 1 lv. + 2 * 2 lv. + 1 * 5 lv. = 10 lv. 3 * 1 lv. + 1 * 2 lv. + 1 * 5 lv. = 10 lv.
5 **3** **1** **7**	0 * 1 lv. + 1 * 2 lv. + 1 * 5 lv. = 7 lv. 1 * 1 lv. + 3 * 2 lv. + 0 * 5 lv. = 7 lv. 2 * 1 lv. + 0 * 2 lv. + 1 * 5 lv. = 7 lv. 3 * 1 lv. + 2 * 2 lv. + 0 * 5 lv. = 7 lv. 5 * 1 lv. + 1 * 2 lv. + 0 * 5 lv. = 7 lv.

26. Even Pairs

Write a program that generates and prints on the console four-digit numbers in which the first and second pairs of digits form two-digit primes (an example of such a number 1723). The final value to which the pairs should be generated is determined by another 2 digits received as input, which determine how much the final value is greater than the initial

The input contains exactly 4 parameters:

- **In the first row** – the initial value of the first **first pair of** numbers – **a positive integer in the range [10… 90]**
- **In the second row** – the initial value of the **second pair of** numbers – **a positive integer in the range [10… 90]**
- **In the third row** – the difference between the initial and final values **of the first** pair of numbers – **a positive integer in the range [1… 9]**
- **In the fourth row** – the difference between the initial and final values of the **second** pair of numbers – **a positive integer in the range [1… 9]**

Print four-digit numbers in which the first two **and** second two digits **are** prime **two-digit numbers.**

Input	Output	Explanations
10 **20** **5** **5**	1123 1323	The initial value of the first pair of digits is 10, and of the second 20. The final values are respectively: **10 + 5 = 15** **20 + 5 = 25** There are the following combinations: **1020 1021 1022 1023 1024 1025 1120 1121 1122 1123 1124 1125 … 1320 1321 1322 1323 1324 1325 1420 1421 1422 1423 1424 1425 1520 1521 1522 15 23 1524 1525** But of these, only **1123 and 1323** are **four-digit numbers in which the first part and the second are** both primes.
10 **30** **9** **6**	1131 1331 1731 1931	

27. Change

Write a function that accepts an amount - the change that needs to be returned and calculates how many coins are needed.

Sample input:

input	output	explanation
1.23	4	Our change is 1 lev and 23 stotinki. 4 coins: 1 lev coin, 20 stotinki coin, 2 stotinki coin and 1 stotinka coin
2	1	Our change is 2 leva. 1 coin of 2 leva.
0.56	3	Our change is 56 cents. 3 coins: a 50 stotinki coin, a 5 stotinki coin, and a 1 stotinka coin.
2.73	5	Our change is 2 leva and 73 stotinki. The machine returns it to us with 5 coins: a 2 leva coin, a 50 stotinki coin, a 20 stotinki coin, a 2 stotinki coin, and a 1 stotinki coin.

28. Pyramid of numbers

Write a function that takes an integer n and prints a pyramid of numbers as in the examples:

Sample input:

input	output

7	1
	2 3
	4 5 6
	7

10	1
	2 3
	4 5 6
	7 8 9 10

15	1
	2 3
	4 5 6
	7 8 9 10
	11 12 13 14 15

29. Unique codes

Write a function that generates three-digit codes, with the digits of each code in a certain interval. For a code to be valid, it must meet the following conditions:

- The first and third digits must be even.
- The second digit must be a prime number in the range [2...7].

Input

You take 3 parameters:

- The upper limit of the first number - an integer in the range [1...9]
- The upper limit of the second number - an integer in the range [1...9]
- The upper limit of the third number - an integer in the range [1...9]

Output

Print on the console all valid three-digit codes whose digits correspond to the appropriate intervals.

Sample input:

input	output

3
5
5
	2 2 2
	2 2 4
	2 3 2
	2 3 4
	2 5 2
	2 5 4

6
2
6
	2 2 2
	2 2 4
	2 2 6
	4 2 2
	4 2 4
	4 2 6
	6 2 2
	6 2 4
	6 2 6

Guidelines

Check online how you can tell if a number is prime? **IsPrime**

30. Square of Asterisks

Write a function that takes a number n and draws a square of n * n asterisks. Between each two asterisks there is a whitespace.

Sample input:

input	output
2	* * * *
3	* * * * * * * * *

31. Half-Rhombus from asterisks

Write a program that accepts a positive integer n and prints a rhombus of asterisks of size n as in the examples below:

Sample input:

input	output
1	*
2	* * *

	*
3	* * * * * * * * *
4	* * * * * * * * * * * * * * * *

32. Rhombus from asterisks

Write a program that accepts a positive integer n and prints a rhombus of asterisks of size n as in the examples below:

Sample input:

input	**output**
1	*
2	* * * *
3	* * * * * * * * *
4	* * * * * * * * * * * * * * * *

33. Tree Pattern

Input: A positive integer 'n' representing the height of the tree.

Input	**Output**

6	*
	    ***
	   *****
	  *******
	 *********
	***********

	`     *`

4	`   *`
	`  ***`
	` *****`
	`*******`
	`   *`

3	`  *`
	` ***`
	`*****`
	`  *`

34. Square frame

Write a program that reads a positive integer **n and draws** a square frame **of size n as in the examples below:**

Input	Output		Input	Output		Input	Output		Input	Output

3	`+ - +`
	`| - |`
	`+ - +`
			4	`+ - - +`
				`| - - |`
				`| - - |`
				`+ - - +`
						5	`+ - - - +`
							`| - - - |`
							`| - - - |`
							`| - - - |`
							`+ - - - +`
									6	`+ - - - - +`
										`| - - - - |`
										`| - - - - |`
										`| - - - - |`
										`| - - - - |`
										`+ - - - - +`

35. Christmas Tree

Write a program that receives a number **n (1** ≤ n ≤ 100) and prints **a Christmas tree** of size n **as in the examples below:**

Input	Output		Input	Output		Input	Output		Input	Output

1	`  |`
	`* | *`
			2	`   |`
				` * | *`
				`** | **`
						3	`    |`
							`  * | *`
							` ** | **`
							`*** | ***`
									4	`     |`
										`   * | *`
										`  ** | **`
										` *** | ***`
										`**** | ****`

36. Sunglasses

Write a program that reads an integer n **(3** ≤ n ≤ 100) and prints **sunglasses** of size **5*n** x n **as in the examples:**

Input	Output

3	`******   ******`
	`*////*|||*////*`
	`******   ******`

4	`********    ********`
	`*//////*||||*//////*`

	`*//////*    *//////*`
	`********    ********`

5	`**********     **********`
	`*////////*     *////////*`
	`*////////*|||||*////////*`
	`*////////*     *////////*`
	`**********     **********`

Tips:

- Print the **top row** of glasses:
  - **2*n** asterisks; **n** spaces; **2*n** asterisks
- Print the **middle n-2 lines**:
  - asterisk; **2*n-2** slashes; asterisk; **n** spaces; asterisk; **2*n-2** slashes; asterisk
  - On row **(n-1) / 2 - 1**, print **n** vertical bars instead of **n** spaces.
- Print the **bottom row** of glasses:
  - **2*n** asterisks; **n** spaces; **2*n** asterisks

37. House Pattern

Input: A positive odd integer 'n' representing the height of the house.

Input	Output

7	`   *`
	`  ***`
	` *****`
	`*******`
	`*     *`
	`*     *`

4	`  * *`
	`* * * *`
	`*     *`
	`*     *`

3	`  *`
	`* * *`
	`*   *`
	`*   *`

38. Pyramid with Increasing Digits

Input: A positive integer 'n' representing the number of rows.

Input	Output

5	`    1`
	`   232`
	`  34543`
	` 4567654`
	`567898765`

4	1
	   232
	  34543
	 4567654

3	1
	   232
	  34543

39. Arrow Pattern

Input: A positive integer 'n' representing the number of rows.

Input	Output

7	*
	    * *
	   * * *
	  * * * *
	 * * * * *
	* * * * * *
	   *****
	   *****
	   *****
	   *****
	   *****
	   *****

4	* * * * * * *** *** ***

6	*
	   * *
	  * * *
	 * * * *
	* * * * *
	  *****
	  *****
	  *****
	  *****
	  *****

40. Staircase Pattern

Input: A positive integer 'n' representing the number of steps.

Input	Output
5	# ## ### #### #####
4	# ## ### ####
2	# ##

41. Hourglass Pattern

Draw an hourglass pattern:

Input	Output
5	####### # # # # # # # # # #######

8	##########
	 # #
	  # #
	   # #
	    ##
	   # #
	  # #
	 # #
	##########

2	#### ## ####

42. Left Arrow Pattern

Draw a left arrow pattern:

Input	Output

5	`   *`
	`  **`
	` ***`
	`****`
	` ***`
	`  **`
	`   *`

4	`  *`
	` **`
	`***`
	` **`
	`  *`

10	`        *`
	`       **`
	`      ***`
	`     ****`
	`    *****`
	`   ******`
	`  *******`
	` ********`
	`*********`
	` ********`
	`  *******`
	`   ******`
	`    *****`
	`     ****`
	`      ***`
	`       **`
	`        *`

43. Pyramid of numbers

Write a function that takes an integer n and prints a pyramid of numbers, as in the examples:

Sample input:

input	output

7	1
	2 3
	4 5 6
	7

10	1
	2 3
	4 5 6
	7 8 9 10

12	1
	2 3
	4 5 6

	7 8 9 10 11 12

44. Alternative conditions

Sample input:

input	output

7	1
	  2 3
	4 5 6
	    7

10	1
	    2 3
	  4 5 6
	7 8 9 10

12	1
	    2 3
	  4 5 6
	7 8 9 10
	   11 12

Sample input:

input	output

7	1
	  2 3
	 4 5 6
	7

10	1
	  2 3
	 4 5 6
	7 8 9 10

12	1
	   2 3
	  4 5 6
	 7 8 9 10
	11 12

45. Equal sum of odd and even positions

Write a function that accepts two six-digit integers ranging from 100000 to 300000. Always the first number entered will be less than the second. On the console print all numbers that are located between the two meeting the following condition:

- the sum of the digits of even and odd positions shall be equal.

If there are no numbers matching the condition, “None” is displayed.

Sample input:

input	output

100000 **100050**	100001 100012 100023 100034 100045
299900 **300000**	299970 299981 299992
100115 **100120**	None

46. Password Generator

Write a program that takes two integers n and l and generates alphabetically all possible passwords that consist of the following 5 characters:

- Symbol 1: digit from 1 to n.
- Symbol 2: digit from 1 to n.
- Symbol 3: lowercase letter among the first l letters of the Latin alphabet.
- Symbol 4: a lowercase letter among the first l letters of the Latin alphabet.
- Symbol 5: a digit from 1 to n, greater than the first 2 digits.

Sample input:

input	output
2 **4**	11aa2 11ab2 11ac2 11ad2 11ba2 11bb2 11bc2 11bd2 11ca2 11cb2 11cc2 11cd2 11da2 11db2 11dc2 11dd2
3 **1**	11aa2 11aa3 12aa3 21aa3 22aa3

47. Special numbers

Write a function that takes an integer n and generates all possible "special" numbers from 1111 to 9999. For a number to be "special," it must meet the following condition:

- n to be divided by each of its digits without a remainder.

For example, n = 16, 2418 is a special number:

- 16 / 2 = 8 without remainder
- 16 / 4 = 4 without remainder
- 16 / 1 = 16 without remainder
- 16 / 8 = 2 without remainder

Sample input:

input	**output**	**input**	**output**
3	1111 1113 1131 1133 1311 1313 1331 1333 3111 3113 3131 3133 3311 3313 3331 3333	2	1111 1112 1121 1122 1211 1212 1221 1222 2111 2112 2121 2122 2211 2212 2221 2222