# Exercises: Nested Loops

In-class and homework exercises for the "**Programming Basics**" course [@ SoftUni Global](https://softuni.org).

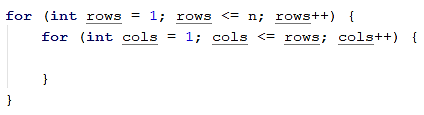
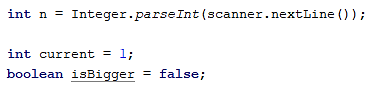
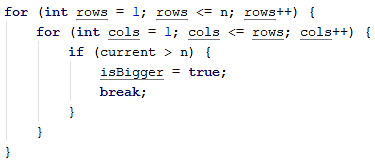
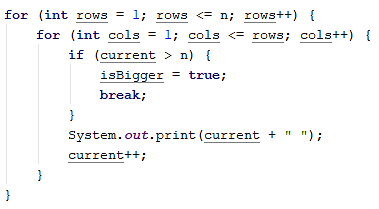
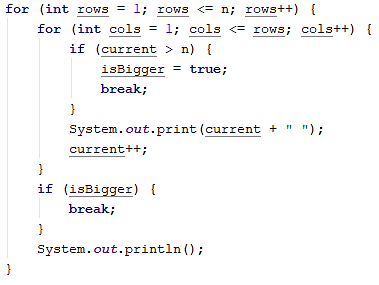
**Test** your solutions in the **judge system**: <https://judge.softuni.org/Contests/Compete/Index/3551>

## Number Pyramid

Write a program, which reads an integer **a**, entered by the user, and prints a **pyramid of numbers** like in the Sample Input and Output:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **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 | 15 | 1  2 3  4 5 6  7 8 9 10  11 12 13 14 15 |

### Hints and Guidelines

1. **Read** one **integer from the console**:  
   
2. **Create two nested for loops,** with which to print the pyramid of numbers, as the outer loop will determine **how many rows** to print, and the inner - **how many numbers** are printed on the corresponding **row**:  
   
3. In a separate **counter** keep how many numbers you have printed **so far** (and what is the current number). When you reach **n**, exit the two nested loops with break**. To get out of both cycles we have to use the break operator in both of them**. For this purpose, we will make a boolean variable to check if we have exited from the internal loop. Go to the beginning of the program and initialize the following two variables:   
   
4. In the internal for loop, check if the current variable has become greater than n. If so, change the value of the boolean variable and exit the internal loop:  
   
5. After the checks, print the variable current in the desired format and increase it by 1. If you have left the loop, printing will not be needed!  
   
6. In the body of the external loop, check whether we need to get out of it too. Then print one blank line so that the next numbers are on a new line. If we have exited from the external loop, we would not get to execute the command System.out.println()! Your program should look like this:  
   

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3551#0>

## Equal Sums Even Odd Position

Write a program, which reads from the console **two six-digit integers** in the range from 100000 to 300000. The **first** number will always be **smaller than the second one.** Print on the console on **1 line separated by an interval,** all numbers which are **between the two numbers** read by the console at the start and which also meet the following **requirement**:

* **The sum** of the numbers of **even** and **odd** positions must be **equal**. If there are no numbers that meet the requirement, no result is printed on the console.

### Sample Input and Output

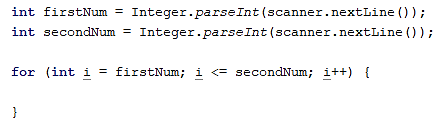
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** | | | |
| 100000  100050 | 100001 100012 100023 100034 100045 | The first number we generate is the number 100000. The sum of the digits of even positions (yellow) is 0 + 0 + 0 = 0. The sum of the digits of the odd positions (green) is 0 + 0 + 1 = 1. Because the two amounts are different, the number is not printed on the console.  The next number is 100001. The sum of even positions is 1 + 0 + 0 = 1, and of odd positions 0 + 0 + 1 = 1. The two sums are equal and the number is printed.  The next number to check is 100002. It doesn’t meet the requirements and is not printed  ……  With the number 100045, the sum of the even positions is 5 + 0 + 0 = 5, and of the odd 4 + 0 + 1 = 5. Both amounts are equal the number is printed. Etc. | | | |
| **Input** | **Output** | **Input** | **Output** | **Input** | **Output** |
| 123456  124000 | 123464 123475 123486 123497 123530 123541 123552 123563 123574 123585 123596 123640 123651 123662 123673 123684 123695 123750 123761 123772 123783 123794 123860 123871 123882 123893 123970 123981 123992 | 299900  300000 | 299970 299981 299992 | 100115  100120 | *No Output* |

### Hints and Guidelines

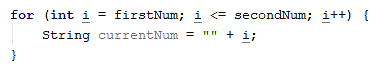
1. Read the input data from the user:



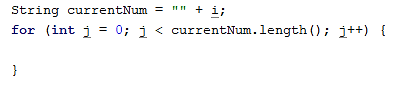
1. To check all the numbers in the interval, create a for loop. After reading the input numbers, set the first number for the initial value of the control variable**.** Iterate until the **second number** is reached by **increasing** the value of the control variable by **1**:

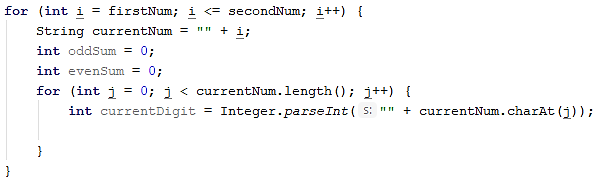


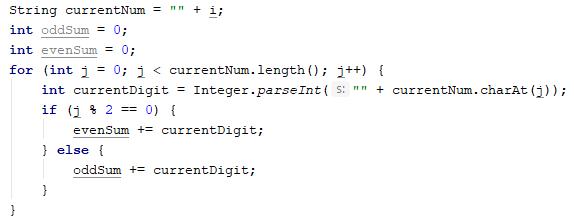
1. Take the number of the current position **as text** by **concatenating it with an empty string**:



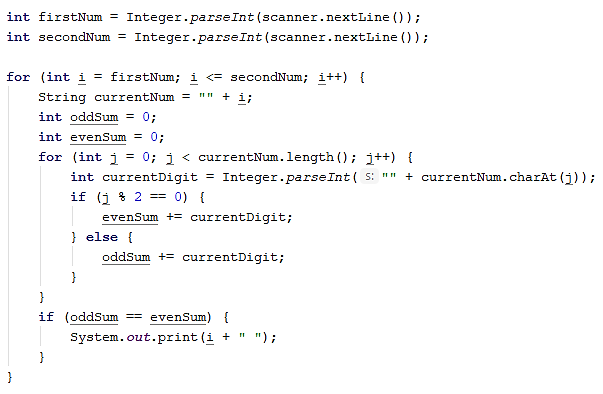
1. Make a **for** loop for going through all digits of the number. Once taken as text, take its length with .length(). Iterate until the **length of the number** is reached by **increasing** the value of the control variable by **1**:



1. Continue to add the logic for finding the sum of even and odd positions for each number. Declare one variable for the even and odd sum. To get the exact value of the digits, use the method Integer.parseInt(). 
2. To find digits that are in an even position, use the conditional **if** construction, check if its index is an even number by dividing it modularly by 2 (**index % 2**), if it is even, add it to the sum of the evens if it is not, to the sum of the odds.



1. Once you have found the sum of the digits of even and odd positions, check if they are equal, and if they are, print the number. Your program should look like this:



### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3551#1>

## Sum Prime Non-Prime

Write a program, which reads integers from the console, until receiving the command "**stop"**. Find the **sum** of all **prime numbers** and the num of all **non-prime numbers.** Due to the mathematical definition that negative numbers cannot be prime, if a **negative number** is received as input, print the following message "**Number is negative.**". In this case, the entered number is ignored and is not added to either of the two sums, and the program continues its execution, waiting for the next number to be entered. **Print** two lines of the **two calculated sums** in the following format:

* "Sum of all prime numbers is: {prime numbers sum}"
* "Sum of all non prime numbers is: {nonprime numbers sum}"

### Sample Input and Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Comments** | |
| 3  9  0  7  19  4  stop | Sum of all prime numbers is: 29  Sum of all non prime numbers is: 13 | The first entered number is 3. It is prime, so we add it to the sum of prime numbers.  The next number is 9. It is not prime, so we add it to the sum of non-prime numbers.  The number 0 is not prime, so we add it to the sum of non-prime numbers. The sum is 9+0=9.  The next two numbers are 7 and 19. Both are prime, so we add them to the sum of prime numbers. 3+7=10 and 10+19=29.  The next number is 4, which is not prime, so we add it to the corresponding sum 9+4=13.  The stop command is received. The program stops its execution and we print both sums. | |
| **Input** | **Output** | **Input** | **Output** |
| 30  83  33  -1  20  stop | Number is negative.  Sum of all prime numbers is: 83  Sum of all non prime numbers is: 83 | 0  -9  0  stop | The number is negative.  Sum of all prime numbers is: 0  Sum of all non prime numbers is: 0 |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3551#2>

1. **Train the Trainers**

The "Train the trainers" course is coming to an end and the final evaluation is approaching. Your task is to help the jury that will evaluate the presentations by writing a program in which to calculate the **average grade** of **each of the student’s presentations**, and finally the **average grade of all of them**.

On the first line of the console, the number of people in the jury is read as **n** – **integer in the range[1…20]**

After that, the name of the presentation is read in a new line - **text**

For each of the presentation, on new lines is read **n – the number of grades – a floating-point number in the range [2.00…6.00]**

**After calculating the average grade for a specific presentation,** print on the console:

**"{presentation\_title} - {average\_grade}."**

After receiving the command **"Finish"** print on the console:

**"Student's final assessment is {average grade from all presentations}."** and the program stops.

All grades must be formatted **two decimal places** after the comma.

### Sample Input and Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Comments** | |
| 2  While-Loop  6.00  5.50  For-Loop  5.84  5.66  Finish | While-Loop - 5.75.  For-Loop - 5.75.  Student's final assessment is 5.75. | 2 – the number of the people in the jury, therefore each presentation will receive 2 grades.  (6.00 + 5.50) / 2 = 5.75  (5.84 + 5.66) / 2 = 5.75  (6.00 + 5.50 + 5.84 + 5.66) / 4 = 5.75 | |
| **Input** | **Output** | **Input** | **Output** |
| 3  Arrays  4.53  5.23  5.00  Lists  5.83  6.00  5.42  Finish | Arrays - 4.92.  Lists - 5.75.  Student's final assessment is 5.34. | 2  Objects and Classes  5.77  4.23  Dictionaries  4.62  5.02  RegEx  2.88  3.42  Finish | Objects and Classes - 5.00.  Dictionaries" - 4.82.  RegEx - 3.15.  Student's final assessment is 4.32. |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3551#3>

## Special numbers

Write a program, which **reads an integer N**, given by the user, and generates all possible **"special"** **numbers** from **1111** to **9999**. In order for a number to be **"special"**, it needs to meet the **following condition**:

* **N to be divisible by each of its digits without remainder.**

**Example:** with **N = 16**, **2418** is a special number:

* **16 / 2** = 8 **no remainder**
* **16 / 4** = 4 **no remainder**
* **16 / 1** = 16 **no remainder**
* **16 / 8** = 2 **no remainder**

### Input Data

The input is received from the console and consists of **one integer** in the range [**1**…**600000**]

### Output Data

Print **all "special" numbers** on the console, separated by an **interval.**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3 | 1111 1113 1131 1133 1311 1313 1331 1333 3111 3113 3131 3133 3311 3313 3331 3333 | 3 / 1 = 3 no remainder  3 / 3 = 1 no remainder  3 / 3 = 1 no remainder  3 / 3 = 1 no remainder |
| 11 | 1111 | |
| 16 | 1111 1112 1114 1118 1121 1122 1124 1128 1141 1142 1144 1148 1181 1182 1184 1188 1211 1212 1214 1218 1221 1222 1224 1228 1241 1242 1244 1248 1281 1282 1284 1288 1411 1412 1414 1418 1421 1422 1424 1428 1441 1442 1444 1448 1481 1482 1484 1488 1811 1812 1814 1818 1821 1822 1824 1828 1841 1842 1844 1848 1881 1882 1884 1888 2111 2112 2114 2118 2121 2122 2124 2128 2141 2142 2144 2148 2181 2182 2184 2188 2211 2212 2214 2218 2221 2222 2224 2228 2241 2242 2244 2248 2281 2282 2284 2288 2411 2412 2414 2418 2421 2422 2424 2428 2441 2442 2444 2448 2481 2482 2484 2488 2811 2812 2814 2818 2821 2822 2824 2828 2841 2842 2844 2848 2881 2882 2884 2888 4111 4112 4114 4118 4121 4122 4124 4128 4141 4142 4144 4148 4181 4182 4184 4188 4211 4212 4214 4218 4221 4222 4224 4228 4241 4242 4244 4248 4281 4282 4284 4288 4411 4412 4414 4418 4421 4422 4424 4428 4441 4442 4444 4448 4481 4482 4484 4488 4811 4812 4814 4818 4821 4822 4824 4828 4841 4842 4844 4848 4881 4882 4884 4888 8111 8112 8114 8118 8121 8122 8124 8128 8141 8142 8144 8148 8181 8182 8184 8188 8211 8212 8214 8218 8221 8222 8224 8228 8241 8242 8244 8248 8281 8282 8284 8288 8411 8412 8414 8418 8421 8422 8424 8428 8441 8442 8444 8448 8481 8482 8484 8488 8811 8812 8814 8818 8821 8822 8824 8828 8841 8842 8844 8848 8881 8882 8884 8888 | |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3551#4>

## Cinema tickets

Your task is to **write a program**, which calculates the **percentage of tickets for each kind of the sold tickets**: student(**student**), standard(**standard**) и kid(**kid**), for all screenings. You also need to calculate **what percentage of the hall is filled** for **each screening**.

### Input Data

The input is a series of **integers** and **text**:

* On the first line, until receiving the "**Finish**" command – Title of the movie– **text**
* On the second line – the free seats in the hall for each screening – **integer [1 … 100]**
* For each movie, read one line until there are no more free seats in the hall, or until the "**End**" command is received:
  + The type of the ticket – text ("**student", "standard", "kid"**)

### Output Data

Print the **following lines** to the console:

* After each movie, print the percentage of taken seats in the hall:

**"{Movie title} - {percentage of taken seats}% full."**

* On receiving the "Finish" command, print for lines:
  + **"Total tickets: {the sum of all sold tickets for all movies}"**
  + **"{percentage of student tickets}% student tickets."**
  + **"{percentage of standard tickets}% standard tickets."**
  + **"{percentage of kid tickets}% kids tickets."**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Taxi  10  standard  kid  student  student  standard  standard  End  Scary Movie  6  student  student  student  student  student  student  Finish | Taxi - 60.00% full.  Scary Movie - 100.00% full.  Total tickets: 12  66.67% student tickets.  25.00% standard tickets.  8.33% kids tickets. | First movie – Taxi, the seats in the hall are 10  3 standard, 2 student, 1 kid tickets are bought until receiving the End command.  There are 6 tickets out of 10 seats -> 60% of the hall is occupied.  Second movie – Scary Movie, the seats in the hall are 6  6 student tickets are bought and there are no more free seats in the hall.  There are 6 tickets out of 6 seats -> 100% of the hall is occupied.  The command Finish is received.  The total number of purchased tickets for all movies is 12.  For all movies, there are:  8 student tickets. 8 tickets out of 12 are 66.67%  3 standard tickets. 3 tickets out of 12 are 25%  1 kid ticket. 1 ticket out of 12 is 8.33% |
| **Input** | **Output** | **Comments** |
| The Matrix  20  student  standard  kid  kid  student  student  standard  student  End  The Green Mile  17  student  standard  standard  student  standard  student  End  Amadeus  3  standard  standard  standard  Finish | The Matrix - 40.00% full.  The Green Mile - 35.29% full.  Amadeus - 100.00% full.  Total tickets: 17  41.18% student tickets.  47.06% standard tickets.  11.76% kids tickets. | First movie – The Matrix, the seats in the hall are 20  2 standard, 4 student, 2 kid tickets are bought until receiving the End command.  There are 8 tickets out of 20 seats -> 41.18% of the hall is occupied.  Second movie - The Green Mile, the seats in the hall are 17  3 standard, 3 student tickets are bought until receiving the End command.  There are 6 tickets out of 17 seats-> 47.06% of the hall is occupied.  Third movie – Amadeus, the seats in the hall are 3  3 standard tickets are bought and there are no more free seats in the hall.  There are 3 tickets out of 3 seats-> 100% от of the hall is occupied.  The command Finish is received.  The total number of purchased tickets for all movies is 17.  For all movies, there are:  7 student tickets. 7 tickets out of 17 are 41.18%  8 standard tickets. 8 tickets out of 17 are 47.06%  2 kid tickets. 2 tickets out of 17 are 11.76% |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3551#5>