# Exercise: Nested Loops

Tasks for exercise in class and for homework to the course ["Programming Fundamentals and Unit Testing" @ SoftUni.](https://softuni.bg/trainings/4256/programming-fundamentals-and-unit-testing-september-2023)

**Test** your solution in **the Judge system**: <https://judge.softuni.org/Contests/4347>

## Number Pyramid

Write a program that reads a whole number n entered by the user and prints a **pyramid of numbers** as shown in the examples:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

### Guidelines

1. **Read** an **integer from the console:**  
   
2. **Create two nested for loops** to print the number pyramid. The outer loop determines **how many rows to print**, and the inner loop determines **how many numbers to print on each row**:  
   
3. Use a separate **counter** to keep track of how many numbers you've printed **so far** (and what the current number is). When you reach **n**, exit both nested loops **using the break statement**. **To exit both loops, you'll need to use a boolean variable that checks if you've exited the inner loop. Go to the beginning of the program and initialize the following two variables:**



1. In the inner for loop, check if the current variable has become greater than n. If it has, change the value of the boolean variable and exit the inner loop:  
   
2. After the check, print the current variable in the desired format and increment it by 1. If you've exited the loop, this print statement won't be reached!  
   
3. Inside the body of the outer loop, check if you need to exit this loop as well. Then, print an empty line to move to the next line for the next set of numbers. If you've exited the outer loop, this Console.WriteLine() command won't be executed! Your program should look like the following structure:



## Equal Sums of Even and Odd Positions

Write a program that reads **two six-digit integers** from the console in the range from 100000 to 300000. **The first entered number will always be smaller than the second one**. **Print on one line, separated by spaces**, all numbers that are **between the two numbers** red from the console and satisfy the following condition:

* **The sum** of the digits **at even and odd positions** must be **equal**. If there are no numbers that meet this condition, do not output any result on the console.

### Example 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 at even positions (in yellow) is 0+0+0=0. The sum of the digits at odd positions (in green) is 0+0+1=1. Since the two sums are different, the number is not printed.  The next number is 100001. The sum of the digits at even positions is 1+0+0=1, and the sum of the digits at odd positions is 0+0+1=1. The two sums are equal, and the number is printed.  The next number for checking is 100002. It does not meet the condition and is not printed.  ……  For the number 100045 the sum of the digits at even positions is 5+0+0=5, and the sum of the digits at odd positions is 4+0+1=5. The two sums are equal, and the number is printed. And so on. | | | |
| **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* |

### Guidelines

1. Read the input data from the user:



1. To go through all the numbers in the interval, create a **for loop**. After reading the input numbers, set the first number as the initial value of the loop variable. Iterate until reaching **the second number by increasing the value** of the loop variable **by 1**:



1. Take the number at the current position **as text, using the method** .ToString():



1. To iterate through each digit of the number, create a **for loop**. After converting the number to text, obtain its length using .Length. Iterate until reaching the **length of the number**, **increasing the value** of the control variable by **1**:



1. Proceed with adding the logic to find the sum of digits at even and odd positions for each number. Declare a variable for both even and odd sums. To obtain the exact numerical value of the digits, make use of the int.Parse() method.



1. To find the digits located at even positions, use a **conditional if statement** to check if their index is an even number by using the **modulo operation with 2 (index % 2)**. If it's even, add the digit to the sum of even positions; otherwise, add it to the sum of odd positions.



1. Once you've calculated the sum of the digits at even and odd positions, check if they are equal. If they are, print the number. Your program should look like this: 

## Sum of Prime and Non-Prime Numbers

Write a program that reads integer numbers from the console until the command "**stop**" is received. Find **the sum** of all entered **prime numbers** and the sum of all entered **non-prime numbers**. As per mathematical definition, negative numbers cannot be prime. If a **negative number is entered**, display the message **"Number is negative."** In this case, ignore the entered number and do not add it to either of the two sums. The program continues its execution, awaiting the input of the next number.

To determine if a number is prime, we can use the following algorithm:

int divisors = 0;

for (int i = 1; i <= number; i++)

{

if(number % i == 0){divisors++;}

}

if(divisors == 2)

//only two divisors, this is a prime number…

Print the **two calculated sums** in the following format **on two separate lines**:

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

### Example 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 a prime number, so we add it to the sum of prime numbers.  The next number is 9It is not a prime number, so we add it to the sum of non-prime numbers.  The number s not a prime number, and we add it to the sum of non-prime numbers. The sum becomes 9 + 0 = 9.  The next two numbers are 7 and 19. Both are prime numbers, and we add each of them to the sum of prime numbers. 3 + 7 = 10 and 10 + 19 = 29.  Next is the number 4, which is not a prime number, so we add it to the corresponding sum: 9+4=13.  We receive the "stop" command. The program terminates its execution, and we print the two calculated 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 | Number is negative.  Sum of all prime numbers is: 0  Sum of all non prime numbers is: 0 |

1. **Train the Trainers**

The "Train the trainers" course is near its end, and the final assessment is approaching. Your task is to assist the jury that will evaluate the presentations by creating a program that calculates **the average grade for each student's presentation and finally the overall average success**.

Read from the console, the number of jury members: **n - an integer in the range [1…20]**

Then, the name of the presentation is read on a separate line: **a string**

For each presentation, **n grades are read on separate lines - real numbers in the range [2.00…6.00]**

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

**"{presentation name} - {average grade}."**

Upon receiving the command **Finish**, print on the console:

**"Student's final assessment is {overall average success}."** and the program finishes.

All grades should be formatted **to two decimal places**.

### Example 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 people in the jury, so you will receive 2 grades per presentation.  (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. |

## Special Numbers

Write a program that reads **an integer N** from the user and generates **all possible "special" numbers from 1111 to 9999**. For a number to be "**special**", it must satisfy the **following condition**:

* + **N should be divisible by each of its digits without a remainder.**

**Example:** For **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**

**HINT***: If a number contains zero as any of its digits, it’s never a special number.*

### Input

The input is read from the console and consists of a **single integer in the range [1…600000]**

### Output

Print **all "special" numbers** on the console, separated by **spaces**

### Example 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 without remainder  3 / 3 = 1 without remainder  3 / 3 = 1 without remainder  3 / 3 = 1 without 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 | |

## Cinema Tickets

Your task is to **write a program** that calculates statistics about movies screened at a cinema. You need to find the percentage **of tickets sold for different ticket types: "student", "standard",** and **"kid"**. You also need to **calculate the percentage of seats occupied for each movie.**

### Input

The input includes a sequence **of integers and text**:

* + On the first line - **the name of the movie** - **string**
  + On the second line - the available seats in the cinema for the specific movie - an **integer in the range [1 … 100]**
  + Next, **types of tickets** are read from the console on separate lines, **until we have received all ticket types or the** **"End" command is received**:
* The **type of ticket** - text (**"student", "standard", "kid"**)
  + After the "**End**" command is received, it might be received **information for another movie** or "**Finish**" command, that will **end the program** and should **return summary** for the tickets sold.

### Output

The **following lines** should be printed on the console:

* + After each movie, print what percentage of the cinema is full:

**"{movie name} - {percentage}% full."**

* + When the command **"Finish"** is received, print four lines:
* **"Total tickets: {total number of purchased tickets for all movies}"**
* **"{percentage of student tickets} % student tickets."**
* **"{percentage of standard tickets} % standard tickets."**
* **"{percentage of kids tickets} % kids tickets."**

### Example 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, there are 10 seats  3 standard tickets, 2 student tickets, 1 child ticket and the command End is received.  Total of 6 tickets out of 10 seats -> 60% of the hall is occupied.  Second movie – Scary Movie, there are 6 seats.  6 student tickets are bought and the seats are finished.  Total of 6 tickets out of 6 seats -> 100% 100% of the hall is occupied.  The command Finish is received.  Total purchased tickets for all movies are 12.  For all movies, the purchased tickets are as follows:  8 student tickets. 8 tickets out of 12 is 66.67%.  3 standard tickets. 3 tickets out of 12 is 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, there are 20 seats.  2 standard, 4 student, and 2 kid tickets are purchased, and then the End command is received.  Total of 8 tickets out of 20 seats -> 41.18% of the hall is occupied.  Second movie - The Green Mile, there are 17 seats.  3 standard and 3 student tickets are purchased, and then the End command is received.  Total of 6 tickets out of 17 seats -> 47.06% of the hall is occupied.  Third movie - Amadeus, there are 3 seats.  3 standard tickets are purchased, and all seats are filled.  Total of 3 tickets out of 3 seats -> 100% of the hall is occupied.  The command Finish is received.  Total purchased tickets for all movies are 17.  For all movies, the purchased tickets are as follows:  7 student tickets. 7 tickets out of 17 is 41.18%.  8 standard tickets. 8 tickets out of 17 is 47.06%.  2 kid tickets. 2 tickets out of 17 is 11.76%. |