**Упражнениe: Повторения с цикли – For-цикъл**

Задачи за упражнение в клас и за домашно към курса ["Основи на програмирането" @ СофтУни](https://softuni.bg/courses/programming-basics).

Тествайте решенията си в Judge системата: [<https://judge.softuni.bg/Contests/2406>#0](https://judge.softuni.bg/Contests/Compete/Index/2405#0)

## Числа до 1000, завършващи на 7

Напишете функция, която отпечатва числата в диапазона **[1…1000]**, които **завършват на 7**.

|  |  |
| --- | --- |
| **вход** | **изход** |
| (няма) | 7  17  27  …  997 |

### Насоки

1. Направете for **цикъл** от 7 до 997 и проверете всяко число дали завършва на 7. Едно число завършва на 7, когато резултатът от **модулното деление на числото и 10 е равен на 7**.



## Таблицата за умножение

Напишете функция, която получава аргумент число от 1 до 10 и принтира таблицата за умножение в конзолата.

### Примерен вход и изход

|  |  |
| --- | --- |
| **вход** | **изход** |
| ("5") | 1 \* 5 = 5  2 \* 5 = 10  3 \* 5 = 15  4 \* 5 = 20  5 \* 5 = 25  6 \* 5 = 30  7 \* 5 = 35  8 \* 5 = 40  9 \* 5 = 45  10 \* 5 = 50 |

## Високосни години

Напишете функция, която получава два аргумента. Първият е високосна година, а вторият произволна година. Отпечатайте в конзолата на нов ред само **високосните години** в интервала между получените две такива.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| ("1908",  "1919") | 1908  1912  1916 | ("2000",  "2011") | 2000  2004  2008 | ("1584",  "1597") | 1584  1588  1592  1596 | ("2020",  "2032") | 2020  2024  2028  2032 |

## Факториел

Напишете функция, която получава число и отпечатва на конзолата резултата от неговия факториел.

Факториел е функция на цяло число, равна на произведението на всички естествени числа, по-малки или равни на него.

Например факториел от 4 означава: 1\*2\*3\*4 = 24.

### Примерен вход и изход

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |
| ("4") | 24 | ("8") | 40320 |

## Преброй думите в съобщението

Напишете функция, която получава съобщение (текст) и проверява дали съобщението може да бъде изпратено.

### Вход

Функцията получава аргумент **текст**.

### Изход

Да се отпечата на конзолата един ред:

* Ако думите са повече от 10:

"The message is too long to be send! Has {count} words."

* Ако са по-малко:

"The message was send successfully!"

### Примерен вход и изход

|  |  |
| --- | --- |
| **вход** | **изход** |
| ("This message has exactly eleven words. One more as it's allowed!") | The message is too long to be send!Has 11 words. |
| **вход** | **изход** |
| ("This message has ten words and you can send it!") | The message was send successfully! |

## Автомобилно състезание

За състезание във Формула 1, инженерите в екипа имат нужда от изчисление, което да им покаже дали горивото, с което зареждат автомобила ще е достатъчно, за да завършат състезанието.

Напишете функция, която получава три аргумента: заредено **гориво**, **разход** на гориво за първата обиколка и брой **обиколки** от състезанието. При всяка следваща обиколка, автомобила изгаря гориво и тежестта му намалява. Така след всяка обиколка разхода на гориво на автомобила **намалява с 0,1л на обиколка.** Ако горивото е достатъчно, за да финишира автомобила, функцията трябва да отпечата съобщение на конзолата. В противен случай трябва да се отпечата номера на обиколката, за която автомобила вече няма да има гориво.

### Вход

* **Литри гориво** - **дробно число** в интервала **[1...100]**
* **Разход в литри** на автомобила **за първата обиколка** – **дробно число** в интервала **[5...20]**
* **Брой обиколки,** за да бъде завършено състезанието- **цяло** **число** в интервала **[1...40]**

### Изход

Да се отпечата на конзолата един ред:

* Ако горивото е достатъчно:

"Congrats! You won the race!"

* Ако е по-малко:

`You are out of fuel in round {round}!`- където round е номера на обиколката, за която автомобила няма да има повече гориво.

### Примерен вход и изход

|  |  |
| --- | --- |
| **вход** | **изход** |
| ("70", "6.5", "10") | Congrats! You won the race! |
| **вход** | **изход** |
| ("50", "4.3", "30") | You are out of fuel in round 11! |

## Примерни изпитни задачи\*\*

## Хистограма\*

Дадени са n **цели числа** в интервала [**1**…**1000**]. От тях някакъв процент p1 са под 200, друг процент p2 са от 200 до 399, друг процент p3 са от 400 до 599, друг процент p4 са от 600 до 799 и останалите p5 процента са от 800 нагоре. Да се напише програма, която изчислява и отпечатва процентите p1, p2, p3, p4 и p5.

**Пример**: имаме n = **20** числа: 53, 7, 56, 180, 450, 920, 12, 7, 150, 250, 680, 2, 600, 200, 800, 799, 199, 46, 128, 65. Получаваме следното разпределение и визуализация:

|  |  |  |  |
| --- | --- | --- | --- |
| **Диапазон** | **Числа в диапазона** | **Брой числа** | **Процент** |
| < 200 | 53, 7, 56, 180, 12, 7, 150, 2, 199, 46, 128, 65 | 12 | p1 = 12 / 20 \* 100 = **60.00**% |
| 200 … 399 | 250, 200 | 2 | p2 = 2 / 20 \* 100 = **10.00**% |
| 400 … 599 | 450 | 1 | p3 = 1 / 20 \* 100 = **5.00**% |
| 600 … 799 | 680, 600, 799 | 3 | p4 = 3 / 20 \* 100 = **15.00**% |
| ≥ 800 | 920, 800 | 2 | p5 = 2 / 20 \* 100 = **10.00**% |

### Вход

На първия ред от входа стои цялото число n (1 ≤ n ≤ 1000) – брой числа. На следващите n **реда** стои **по едно** **цяло число** в интервала [**1**…**1000**] – числата върху които да бъде изчислена хистограмата.

### Изход

Да се отпечата на конзолата **хистограмата** – **5 реда**, всеки от които съдържа число между 0% и 100%, с точност две цифри след десетичната точка, например 25.00%, 66.67%, 57.14%.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| (["**3",**  "1",  "2",  "999"]) | 66.67%  0.00%  0.00%  0.00%  33.33% | (["**7**",  "800",  "801",  "250",  "199",  "399",  "599",  "799"]) | 14.29%  28.57%  14.29%  14.29%  28.57% | (["**9**",  "367",  "99",  "200",  "799",  "999",  "333",  "555",  "111",  "9"]) | 33.33%  33.33%  11.11%  11.11%  11.11% | (["**14**",  "53",  "7",  "56",  "180",  "450",  "920",  "12",  "7",  "150",  "250",  "680",  "2",  "600",  "200"]) | 57.14%  14.29%  7.14%  14.29%  7.14% |

## Деление без остатък\*

Дадени са n-на брой **цели числа** в интервала [**1**…**1000**]. От тях някакъв **процент** p1 **се делят без остатък на 2**, друг **процент** p2 се **делят без остатък на 3**, друг **процент** p3 се **делят без остатък на 4**. Да се напише програма, която изчислява и отпечатва процентите p1, p2 и p3.

**Пример**: имаме n = **10** числа: 680, 2, 600, 200, 800, 799, 199, 46, 128, 65. Получаваме следното разпределение и визуализация:

|  |  |  |  |
| --- | --- | --- | --- |
| **Деление без остатък на:** | **Числа в диапазона** | **Брой числа** | **Процент** |
| 2 | 680, 2, 600, 200, 800, 46, 128 | 7 | p1 = 7.0 / 10 \* 100 = **70.00**% |
| 3 | 600 | 1 | p2 = 1 / 10 \* 100 = **10.00**% |
| 4 | 680, 600, 200, 800, 128 | 5 | p3 = 5 / 10 \* 100 = **50.00**% |

### Вход

На първия ред от входа стои цялото число n (1 ≤ n ≤ 1000) - брой числа. На следващите n **реда** стои **по едно** **цяло число** в интервала [**1**…**1000**] - числата които да бъдат проверени на колко се делят.

### Изход

Да се отпечатат на конзолата **3 реда**, всеки от които съдържа процент между 0% и 100%, с точност две цифри след десетичната точка, например 25.00%, 66.67%, 57.14%.

* На **първият ред** - процентът на числата които **се делят на 2**
* На **вторият ред** - процентът на числата които **се делят на** **3**
* На **третият ред** - процентът на числата които **се делят на 4**

### Примерен вход и изход

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| (["**10**",  "680",  "2",  "600",  "200",  "800",  "799",  "199",  "46",  "128",  "65"]) | 70.00%  10.00%  50.00% | (["**3**",  "3",  "6",  "9"]) | 33.33%  100.00%  0.00% |

## Заплата\*

**Шеф на компания забелязва че все повече служители прекарват време в сайтове, които ги разсейват.**

**За да предотврати това, той въвежда изненадващи проверки на отворените табове на браузъра на служителите си. Според сайта се налагат различни глоби:**

* **"Facebook" -> 150 лв.**
* **"Instagram" -> 100 лв.**
* **"Reddit" -> 50 лв.**

**От конзолата се четат два реда:**

* **Брой отворени табове в браузъра** **n -** **цяло число в интервала [1...10]**
* **Заплата - число в интервала [700...1500]**

**След това n – на брой пъти се чете име на уебсайт – текст**

**Ако по време на проверката заплатата стане по-малка или равна на 0 лева, на конзолата се изписва   
"You have lost your salary." и програмата приключва. В противен случай след проверката на конзолата се изписва остатъкът от заплатата (да се изпише като цяло число).**

### Примерен вход и изход

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| (["10",  "750",  "Facebook",  "Dev.bg",  "Instagram",  "Facebook",  "Reddit",  "Facebook",  "Facebook"]) | You have lost your salary. | Има 10 отворени таба в браузъра.  Заплатата е 750  За първия таб -> Facebook глоба 150 лв.(750 – 150 = 600)  За втория таб -> Dev.bg не глобяват  За третия таб -> Instagram глоба 100 лв.(600 – 100 = 500)  За четвъртия таб -> Facebook глоба 150 лв.(500 – 150 = 350)  За петия таб -> Reddit глоба 50 лв. (350 – 50 = 300)  За шестия таб -> Facebook глоба 150 лв.(300 – 150 = 150)  За седмия таб -> Facebook глоба 150 лв.(150 – 150 = 0)  Заплатата е равна на 0, следователно се изписва съответният изход и програмата приключва. | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| (["3",  "500",  "Github.com",  "Stackoverflow.com",  "softuni.bg"]) | 500 | (["3",  "500",  "Facebook",  "Stackoverflow.com",  "softuni.bg"]) | 350 |

## Най-малко число\*

Напишете програма, която получава n-на брой **числа** (n > 0) и намира **най-малкото** измежду тях. Първо се получава число n, а след това самите n числа.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| (["2",  "100",  "99"]) | 99 | (["3",  "-10",  "20",  "-30"]) | -30 | (["4",  "45",  "-20",  "7",  "99"]) | -20 | (["1",  "999"]) | 999 | (["2",  "-1",  "-2"]) | -2 |