# Упражнения: Прости проверки

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

## Празен файл

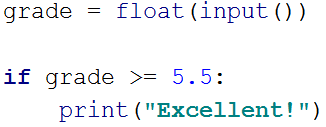
Създайте празен файл

## Проверка за отлична оценка

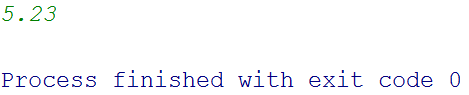
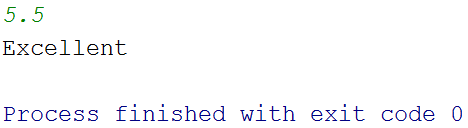
Първата задача от тази тема е да се напише **конзолна програма**, която **въвежда оценка** (десетично число) и отпечатва “**Excellent!**”, ако оценката е **5.50** или по-висока.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 6 | Excellent! | 5 | *(няма изход)* | 5.50 | Excellent! | 5.49 | *(няма изход)* |

1. Създайте **нов файл**
2. Задайте име “Excellent-Result”:
3. Отидете в **началото на файла** и напишете решението на задачата. Можете да си помогнете с кода от картинката по-долу:



1. **Стартирайте** програмата с [Alt+Shift+F10] и я **тествайте** с различни входни стойности:

1. **Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/152#0>. Трябва да получите **100 точки** (напълно коректно решение):



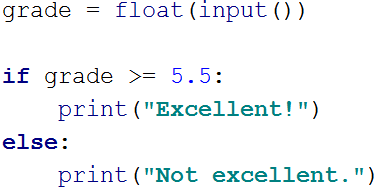


## Отлична оценка или не

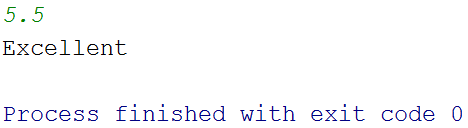
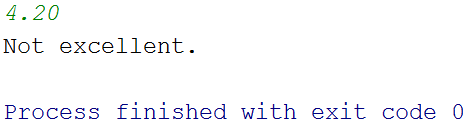
Следващата задача от тази тема е да се напише **конзолна програма**, която **въвежда оценка** (десетично число) и отпечатва “**Excellent!**”, ако оценката е **5.50** или по-висока, или “**Not excellent.**” в противен случай.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 6 | Excellent! | 5 | Not excellent. | 5.50 | Excellent! | 5.49 | Not excellent. |

1. Първо създайте **нов файл,** задайте му име “Excellent-or-Not”.
2. **Напишете кода** на програмата. Може да си помогнете с примерния код от картинката:



1. Сега **стартирайте програмата**, както обикновено с [Alt+Shift+F10] и я тествайте:

1. Тествайте в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#1>. Решението би трябвало да бъде прието като напълно коректно:

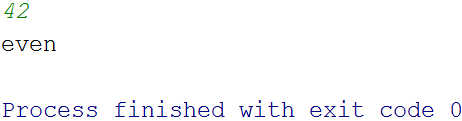


## Четно или нечетно

Да се напише програма, която въвежда **цяло число** и печата дали е **четно** или **нечетно**. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | even | 3 | odd | 25 | even | 1024 | odd |

1. Първо добавете **нов файл.**
2. **Напишете кода** на програмата. Проверката за четност може да се реализира чрез проверка на **остатъка при деление на 2** по следния начин: even = (num % 2 == 0).
3. **Стартирайте** програмата с **[Alt+Shift+F10]** и я тествайте:



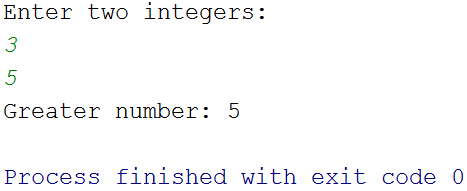
1. Тествайте в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#2>.

## Намиране на по-голямото число

Да се напише програма, която въвежда **две цели числа** и отпечатва по-голямото от двете. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **Изход** |  | **вход** | **изход** |
| 5  3 | 5 | 3  5 | 5 | 10  10 | 10 | -5  5 | 5 |

1. Първо добавете **нов файл**.
2. **Напишете кода** на програмата. Необходима е единична if-else конструкция.
3. **Стартирайте** програмата с **[Alt+Shift+F10]** и я тествайте:



1. Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#3>.

## Изписване на число до 10 с думи

Да се напише програма, която въвежда **цяло число в диапазона [0…10]** и го **изписва с думи** на английски език. Ако числото е извън диапазона, изписва “**number too big**”. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 5 | five | 1 | one | 9 | nine | 10 | number too big |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#4>.

**Подсказка**: можете да напишете дълга if-elif-**el**if-elif…else, с която да разгледате възможните **11 случая**.

## Бонус точки

Дадено е **цяло число** – брой точки. Върху него се начисляват **бонус точки** по правилата, описани по-долу. Да се напише програма, която пресмята **бонус точките** за това число и **общия брой точки** с бонусите.

* Ако числото е **до 100** включително, бонус точките са **5**.
* Ако числото е **по-голямо от 100**, бонус точките са **20%** от числото.
* Ако числото е **по-голямо от 1000**, бонус точките са **10%** от числото.
* Допълнителни бонус точки (начисляват се отделно от предходните):
  + За **четно** число 🡪 + 1 т.
  + За число, което **завършва на 5** 🡪 + 2 т.

Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 20 | 6  26 | 175 | 37  212 | 2703 | 270.3  2973.3 | 15875 | 1589.5  17464.5 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#5>.

**Подсказка**:

* Основните бонус точки можете да изчислите с if-elif-elif-else конструкция (имате 3 случая).
* Допълнителните бонус точки можете да изчислите с if-elif-else конструкция (имате още 2 случая).

## Сумиране на секунди

Трима спортни състезатели финишират за някакъв **брой секунди** (между **1** и **50**). Да се напише програма, която въвежда времената на състезателите и пресмята **сумарното им време** във формат "минути:секунди". Секундите да се изведат с **водеща нула** (2 🡪 "02", 7 🡪 "07", 35 🡪 "35"). Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 35  45  44 | 2:04 | 22  7  34 | 1:03 | 50  50  49 | 2:29 | 14  12  10 | 0:36 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#6>.

**Подсказка**:

* Сумирайте трите числа и получете резултата в секунди. Понеже **1 минута = 60 секунди**, ще трябва да изчислите броя минути и броя секунди в диапазона от 0 до 59.
* Ако резултатът е между 0 и 59, отпечатайте 0 минути + изчислените секунди.
* Ако резултатът е между 60 и 119, отпечатайте 1 минута + изчислените секунди минус 60.
* Ако резултатът е между 120 и 179, отпечатайте 2 минути + изчислените секунди минус 120.
* Ако секундите са по-малко от 10, изведете водеща нула преди тях.

## Конвертор за мерни единици

Да се напише програма, която **преобразува разстояние** между следните 8 **мерни единици**: m, mm, cm, mi, in, km, ft, yd. Използвайте съответствията от таблицата по-долу:

|  |  |
| --- | --- |
| **входна единица** | **изходна единица** |
| **1** meter (**m**) | **1000** millimeters (**mm**) |
| **1** meter (**m**) | **100** centimeters (**cm**) |
| **1** meter (**m**) | **0.000621371192** miles (**mi**) |
| **1** meter (**m**) | **39.3700787** inches (**in**) |
| **1** meter (**m**) | **0.001** kilometers (**km**) |
| **1** meter (**m**) | **3.2808399** feet (**ft**) |
| **1** meter (**m**) | **1.0936133** yards (**yd**) |

Входните данни се състоят от три реда:

* Първи ред: число за преобразуване
* Втори ред: входна мерна единица
* Трети ред: изходна мерна единица (за резултата)

Примерен вход и изход:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 12  km  ft | 39370.0788 ft | 150 mi in | 9503999.99393599 mi | 450  yd  km | 0.41147999937455 yd |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#7>.

## Познай паролата

Да се напише програма, която **въвежда парола** (един ред с произволен текст) и проверява дали въведеното **съвпада** с фразата “s3cr3t!P@ssw0rd”.При съвпадение да се изведе “**Welcome**”. При несъвпадение да се изведе “**Wrong password!**”. Примери:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| qwerty | Wrong password! | s3cr3t!P@ssw0rd | Welcome | s3cr3t!p@ss | Wrong password! |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#8>.

**Подсказка**: използвайте if-else конструкцията.

## Число от 100 до 200

Да се напише програма, която **въвежда цяло число** и проверява дали е **под 100**, **между 100 и 200** или **над 200**. Да се отпечатат съответно съобщения като в примерите по-долу:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 95 | Less than 100 | 120 | k Between 100 and 200 | 210 | Greater than 200 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#9>.

**Подсказка**: използвайте if-elif-elif-else конструкция за да проверите всеки от трите случая.

## Еднакви думи

Да се напише програма, която **въвежда две думи** и проверява дали са еднакви. Да не се прави разлика между главни и малки думи. Да се изведе “yes” или “no”. Примери:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **Вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| Hello  Hello | yes | SoftUni  Softuni | yes | Soft  Uni | no | beer  vodka | no | HeLlO  hELLo | yes |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#10>.

**Подсказка**: използвайте if-else конструкция. Преди сравняване на думите ги обърнете в долен регистър: word = word.lower().

## Информация за скоростта

Да се напише програма, която **въвежда скорост** (десетично число)и отпечатва **информация за скоростта**. При скорост **до 10** (включително) отпечатайте “**slow**”. При скорост **над 10** и **до 50** отпечатайте “**average**”. При скорост **над 50** и **до 150** отпечатайте “**fast**”. При скорост **над 150** и **до 1000** отпечатайте “**ultra fast**”. При по-висока скорост отпечатайте “**extremely fast**”. Примери:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 8 | slow | 49.5 | average | 126 | fast | 160 | ultra fast | 3500 | extremely fast |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#11>.

**Подсказка**: използвайте серия от if-else-if-else**-**… конструкции, за да хванете всичките 5 случая.

## Лица на фигури

Да се напише програма, която **въвежда размерите на геометрична** фигура и пресмята лицето й. Фигурите са четири вида: квадрат (**square**), правоъгълник (**rectangle**), кръг (**circle**) и триъгълник (**triangle**). На първия ред на входа се чете вида на фигурата (square, rectangle, circle или triangle). Ако фигурата е **квадрат**, на следващия ред се чете едно число – дължина на страната му. Ако фигурата е **правоъгълник**, на следващите два реда четат две числа – дължините на страните му. Ако фигурата е **кръг**, на следващия ред чете едно число – радиусът на кръга. Ако фигурата е **триъгълник**, на следващите два реда четат две числа – дължината на страната му и дължината на височината към нея. Резултатът да се закръгли до **3 цифри след десетичната точка**. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| square  5 | 25 | rectangle  7  2.5 | 17.5 | circle  6 | 113.097 | triangle  4.5  20 | 45 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#12>.

**Подсказка**: използвайте серия от if-else-if-else**-**… конструкции, за да обработите 4-те вида фигури.

## Време + 15 минути

Да се напише програма, която **въвежда час и минути** от 24-часово денонощие и изчислява колко ще е **часът след 15 минути**. Резултатът да се отпечата във формат hh:mm. Часовете винаги са между 0 и 23, а минутите винаги са между 0 и 59. Часовете се изписват с една или две цифри. Минутите се изписват винаги с по две цифри, с **водеща нула** когато е необходимо. Примери:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 1  46 | 2:01 | 0  01 | 0:16 | 23  59 | 0:14 | 11  08 | 11:23 | 12  49 | 13:04 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#13>.

**Подсказка**: добавете 15 минути и направете няколко проверки. Ако минутите надвишат 59, увеличете часовете с 1 и намалете минутите със 60. По аналогичен начин разгледайте случая, когато часовете надвишат 23. При печатането на минутите проверете за водеща нула.

## Еднакви 3 числа

Три еднакви числа: да се въведат 3 числа и да се отпечата дали са еднакви (yes / no).

## \*Изписване на число до 100 с думи

Да се напише програма, която превръща число [0…100] в текст: 25 🡪 “twenty five”

# Изпитни задачи от минали издания на курса

## \*Цена за транспорт

*Втора задача от междинния изпит на 6 март 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/169#1)*.*

Студент трябва да пропътува n **километра**. Той има избор измежду **три вида транспорт**:

* **Такси**. Начална такса: **0.70** лв. Дневна тарифа: **0.79** лв. / км. Нощна тарифа: **0.90** лв. / км.
* **Автобус**. Дневна / нощна тарифа: **0.09** лв. / км. Може да се използва за разстояния минимум **20** км.
* **Влак**. Дневна / нощна тарифа: **0.06** лв. / км. Може да се използва за разстояния минимум **100** км.

Напишете програма, която въвежда броя километри n и период от деня (ден или нощ) и изчислява **цената на най-евтиния транспорт**.

### Вход

От конзолата се четат **два реда**:

* Първият ред съдържа числото n – брой километри – цяло число в интервала [1…5000].
* Вторият ред съдържа дума “day” или “night” – пътуване през деня или през нощта.

### Изход

Да се отпечата на конзолата **най-ниската цена** за посочения брой километри.

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 5  day | 4.65 | Разстоянието е под 20 км 🡪 може да се ползва само **такси**. Началната такса е 0.70 лв. Понеже е през деня, тарифата е 0.79 лв. / км. С такси **цената** е: 0.70 + 5 \* 0.79 = **4.65** лв. |
| 7  night | 7 | Разстоянието е под 20 км 🡪 може да се ползва само **такси**. Началната такса е 0.70 лв. Понеже е през нощта, тарифата е 0.90 лв. / км. С такси **цената** е: 0.70 + 7 \* 0.90 = **7.00** лв. |
| 25  day | 2.25 | Разстоянието е над 20 км 🡪 може да се ползва **автобус**, но не може да се ползва влак. Автобусът е най-евтиния възможен вариант. С автобус **цената** е: 25 \* 0.09 = **2.25** лв. |
| 180  night | 10.8 | Разстоянието е над 100 км 🡪 може да се ползва **влак**. Влакът е най-евтиният възможен вариант за пътуване. С влак **цената** е: 180 \* 0.06 = **10.80** лв. |

## \*Тръби в басейн

*Втора задача от междинния изпит на 26 март 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/179#1)*.*

Басейн с **обем V** има **две тръби** от които се пълни. **Всяка тръба има определен дебит** (литрите вода минаващи през една тръба за един час). Работникът **пуска тръбите едновременно** и излиза за **N часа**. Напишете програма, която изкарва състоянието на басейна, **в момента, когато работникът се върне.**

### Вход

От конзолата се четат **четири реда**:

* Първият ред съдържа числото V **– Обем на басейна в литри** – цяло число в интервала [1…10000].
* Вторият ред съдържа числото **P1 – дебит на първата тръба за час** – цяло число в интервала [1…5000].
* Третият ред съдържа числото **P2 – дебит на втората тръба за час**– цяло число в интервала [1…5000].
* Четвъртият ред съдържа числото **H – часовете които работникът отсъства** – число с плаваща запетая в интервала [1.0…24.00]

### Изход

Да се отпечата на конзолата **едно от двете възможни състояния**:

* До колко се е запълнил басейна и коя тръба с колко процента е допринесла. Всички проценти се свеждат до цяло число (без закръгляне).
  + "The pool is **[x]%** full. Pipe 1: **[y]%**. Pipe 2: **[z]%**."

### Aко басейнът се е препълнил – с колко литра е прелял за даденото време, число с плаваща запетая

* + "For **[x]** hours the pool overflows with **[y]** liters."

\* **Имайте предвид**, че поради **свеждането до цяло число** се **губят данни** и нормално **сборът на процентите да е 99%, а не 100%.**

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 1000  100  120  3 | The pool is 66% full. Pipe 1: 45%. Pipe 2: 54%. | За 3 часа:  Първата тръба е напълнила – 300 л.  Втората тръба е напълнила – 360 л.  Общо – 660 л. < 1000 л. => 66% са запълнени  Първата тръба е допринесла с 45% (300 от 660 л.).  Втората тръба е допринесла с 54% (360 от 660 л.). |
| 100  100  100  2.5 | For 2.5 hours the pool overflows with 400 liters. | За 2.5 часа:  Първата тръба е напълнила – 250 л.  Втората тръба е напълнила – 250 л.  Общо – 500 л. > 100 л. => 400 л. са преляли. |

## \*Поспаливата котка Том

*Втора задача от междинния изпит на 24 април 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/181#1)*.*

**Котката Том** обича по цял ден да спи, за негово съжаление стопанинът му си играе с него винаги когато има свободно време. За да се наспи добре, **нормата за игра** на Том е **30 000 минути в година.** Времето за игра на Том **зависи от почивните дни на стопанина му**:

* Когато е на **работа**, стопанинът му си играе с него **по 63 минути на ден**.
* Когато **почива**, стопанинът му си играе с него **по 127 минути на ден.**

Напишете програма, която въвежда **броя почивни дни** и отпечатва дали **Том може да се наспи добре** и колко е **разликата от нормата** за текущата година, като приемем че **годината има 365 дни.**

**Пример**: 20 почивни дни -> работните дни са 345 (365 – 20 = 245). Реалното време за игра е 24 275 минути (345 \* 63 + 20 \*127). Разликата от нормата е 5 725 минути (30 000 – 24 275 = 5 725) или 95 часа и 25 минути.

### Вход

Входът се чете от конзолата и се състои от **едно число – броят почивни дни** – **цяло число** в интервала **[0...365]**

### Изход

На конзолата трябва да се отпечатат **два реда**.

* Ако времето за игра на Том **е над нормата** за текущата година:
  + **На първия ред** отпечатайте: “Tom will run away”
  + **На втория ред** отпечатайте разликата от нормата във формат:

“{H} hours and {M} minutes more for play”

* Ако времето за игра на Том **е под нормата** за текущата година:
  + **На първия ред** отпечатайте: “Tomsleeps well”
  + **На втория ред** отпечатайте разликата от нормата във формат:

“{H} hours and {M} minutes less for play”

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **вход** | **изход** | **коментари** |
| 20 | Tom sleeps well  95 hours and 25 minutes less for play | Почични дни: 20 \* 127 = 2 540 минути игра  Работни дни: 365 - 20 = 345 \* 63 = 21 735 минути игра  30 000 > 24 274 => остават 5725 мин = 95 часа и 25 мин |
| 113 | Tom will run away  3 hours and 47 minutes more for play | Почични дни: 113 \* 127 = 14 351 минути  Работни дни: 365 - 113 = 252 \* 63 = 15 876 минути  30 000 < 30 227 => 227 мин повече = 3 часа и 47 мин |

## \*Реколта

*Втора задача от междинния изпит на 17 юли 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/233#1)*.*

От **лозе с площ X квадратни метри** се заделя **40% от реколтата за производство на вино**. От **1 кв.м лозе** се **изкарват Y килограма грозде**. За **1 литър вино** са **нужни 2,5 кг. грозде**. **Желаното количество вино** за продан е **Z литра**.

Напишете **програма**, която **пресмята колко вино може да се произведе** и **дали** това количество **е достатъчно.** **Ако е достатъчно**, **остатъкът се разделя по равно** **между работниците на лозето**.

### Вход

Входът се чете от конзолата и се състои от **точно 4 реда:**

* 1ви ред: **X кв.м е лозето** – **цяло число в интервала [10 … 5000]**;
* 2ри ред: **Y грозде за един кв.м** – **реално число в интервала [0.00 … 10.00]**;
* 3ти ред: **Z нужни литри вино** – **цяло число в интервала [10 … 600];**
* 4ти ред: **брой работници** – **цяло число в интервала [1 … 20];**

### Изход

На конзолата трябва да се отпечата следното:

* Ако **произведеното** вино е **по-малко от нужното**:
  + “It will be a tough winter! More {недостигащо вино} liters wine needed.”
    - Резултатът трябва да е закръглен към по-ниско цяло число
* Ако **произведеното** вино е **повече от нужното**:
  + “Good harvest this year! Total wine: {общо вино} liters.”
    - Резултатът трябва да е **закръглен към по**-ниско цяло число
  + “{Оставащо вино} liters left -> {вино за 1 работник} liters per person.”
    - И двата резултата трябва да са закръглени към по-високото цяло число

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **вход** | **изход** | **коментари** |
| 650  2  175  3 | Good harvest this year! Total wine: 208 liters.  33 liters left -> 11 liters per person. | **Общо грозде**: 650 \* 2 = **1 300**  **Вино** = 40% \* 1300 / 2,5 = **208**  **208 > 175**  208 - 175 = **33 л остават** -> **11 л на човек** |
| 1020  1.5  425  4 | It will be a tough winter! More 180 liters wine needed. | **Общо грозде**: 1 020 \* 1.5 = **1 530**  **Вино** = 40% \* 1 530 / 2,5 = **244.80**  **244.80 < 425**  425 - 244.8 = **180.2 -> 180** **л не достигат** |

## \*Фирма

*Втора задача от междинния изпит на 28 август 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/274#1)*.*

Фирма получава заявка за изработването на проект, за който са необходими определен брой часове. Фирмата разполага с определен брой дни. През 10% от дните служителите са на обучение и не могат да работят по проекта. Един нормален работен ден във фирмата е 8 часа. Всеки служител може да работи по проекта в извънработно време по 2 часа на ден.

Часовете трябва да са закръглени към по-ниско цяло число (Например –> **6.98** **часа** се закръглят на **6 часа**).

Напишете програма, която изчислява дали **фирмата може да завърши проекта навреме** и **колко часа не достигат или остават**.

### Вход

Входът се чете от **конзолата** и съдържа **точно 3 реда**:

* На **първия** ред са **необходимите** **часовете** – **цяло число в интервала [0 ... 200 000]**
* На **втория** ред са **дните, с които фирмата разполага** – **цяло число в интервала [0 ... 20 000]**
* На **третия** ред е **броят на служителите, работещи извънредно** – **цяло число в интервала [0 ... 200]**

### Изход

Да се **отпечата** на конзолата **един ред**:

* Ако **времето е достатъчно**:
  + “**Yes!{оставащите часове} hours left.**”
* Ако  **времето НЕ Е достатъчно**:
  + “**Not enough time!{недостигащите часове} hours needed.**“

### Примерен вход и изход

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 90  7  3 | Yes!2 hours left. | За проекта са нужни **90 часа**.  Фирмата разполага със **7 дена**.  **10%** от които отиват за обучение, следователно часовете за работа са: 6.3 \* 8 = **50.4 часа**.  **3 служители** **работят извънредно** – 3 \* (2 часа за 7 дена) = **42 часа**.  **Общо часове** = 50.4 + 42 = **92.4 часа -> 92 часа > 90**  Проектът **може да бъде завършен на време** и остават **2 часа**. | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 99  3  1 | Not enough time!72 hours needed. | 50  5  2 | Yes!6 hours left. |