# Типове данни - допълнителни задачи

Практически упражнения към курса [**"Programming Fundamentals" за ученици**](https://github.com/BG-IT-Edu/School-Programming/tree/main/Courses/Applied-Programmer/Programming-Fundamentals).

Тествайте задачите от тази тема в judge: [https://judge.softuni.bg/Contests/2651](https://judge.softuni.bg/Contests/2651%20)

## Граници на типа

Напишете програма, която получава **числен тип** (**като низ**) и отпечатва **максималната** и **минималната стойност** на съответния тип. Ще получите един от следните типове: “int”, “uint”, “long”, “byte” и “sbyte”.

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| int | 2147483647  -2147483648 | byte | 255  0 |

### Подсказки

* Следвайте идеята от този код:  
  

## Проверка на число

Напишете програма, която проверява дали дадено **число** е **цяло** или **с плаваща запетая** и **изведете** “**floating-point**” или “**integer**”, според случая. Ще бъдат въвеждани **само**  **числа**.

### Ограничения

* **Целите числа** ще са в интервала **[-9223372036854775808…9223372036854775807]**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| 3 | integer | 2.31 | floating-point |

## Преливане на вода

Имате **съд за вода** с капацитет от **255 литра**. На следващите **n** реда, ще получите **литри вода**, които трябва да **налеете** във вашия **съд**. Ако капацитета на вашия съд **не е**  достатъчен, изведете “Insufficient capacity!” и **продължете** със следващия ред. На последния ред, изведете **литрите** в **съда**.

### Вход

**Входът** ще се състои от 2 реда:

* На **първи ред**, ще получите **n** – брой **редове**, които ще следват
* От следващите **n реда** – ще получите **количествата** вода, които ще трябва да наливате в съда

### Изход

Всеки път когато нямате достатъчно капацитет в съда, **извеждайте**:

Insufficient capacity!

На последния ред, **изведете** само **литрите** в **съда**.

### Ограничения

* **n** ще е в интервала **[1…20]**
* **liters** ще е в интервала **[1…1000]**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **5**  20  100  100  100  20 | Insufficient capacity!  240 | **1**  1000 | Insufficient capacity!  0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **7**  10  20  30  10  5  10  20 | 105 | **4**  250  10  20  40 | Insufficient capacity!  Insufficient capacity!  Insufficient capacity!  250 |

## Туристическа информация

Напишете програма, която помага на туристите да **преобразуват империални мерки** към метричната система. Вашата програма трябва да поддържа **следните преобразувания**: **мили** към **километри**, **инчове** към **сантиметри**, **футове** към **сантиметри**, **ярдове** към **метри** и **галони** към **литри**. **Таблицата за преобразуване е**:

|  |  |  |
| --- | --- | --- |
| **Имаме:** | **Умножаваме по:** | **Получаваме** |
| miles | 1.6 | kilometers |
| inches | 2.54 | centimeters |
| feet | 30 | centimeters |
| yards | 0.91 | meters |
| gallons | 3.8 | liters |

### Вход

Входът се състои от **два реда**:

* На **първи ред**, ще получите **мярка от имперската система**, която трябва да преобразувате
* На **втори ред**,ще получите **стойността**, която трябва да **преобразувате**

### Изход

Изведете отговора в следния формат:

{начална стойност } {начална мярка} = {преобразувана стойност } {преобразувана мярка }

**Форматирайте** **преобразувана стойност** до **2ри знак след запетаята**.

Изведете началната стойност така както е дадена.

### Ограничения

* Стойността, която трябва да бъде преобразувана ще бъде в интервала [±1.5×10-45… ±3.4×1038]**.**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| miles  12.313 | 12.313 miles = 19.70 kilometers |  | gallons  12 | 12 gallons = 45.60 liters |

## Прогноза за времето

Изобретили сте нова технология за **прогнозиране на времето**, чрез **нумерология**. Ще получите число, чрез което може да прогнозирате времето утре**.** Системата работи по следния начин:

* Ако числото се побира в **sbyte** – времето е­**­ “Sunny**”
* Ако числото се побира в **int** – времето е “**Cloudy**”
* Ако числото се побира в **long** – времето е “**Windy**”
* Ако числото е с плаваща запетая – времето е “**Rainy**”

**Винаги извеждайте най-малкия възможен вариант**.

### Вход

* На първи ред, ще получите число.

### Изход

**Изведете** вашата прогноза за времето.

### Ограничения

* Всяко цяло число ще бъде в интервала **[-9223372036854775808…9223372036854775807]**.

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| 120 | Sunny |  | -1.31 | Rainy |

## Дръж крадеца

В бъдещето, много опасен крадец е избягал. Вашата мисия е да го хванете, но знаете само типа на неговото числено **ID**.

На **първи ред,** ще получите **типа** на **ID-то на крадеца**. На **втори ред,** ще получите **n** – броят на ID-та. Човекът, който има ID **най-близко** до максималната стойност на дадения тип **без да го препълва** е крадецът.

### Вход

* На първи ред, ще получите типа на ID-то на крадеца. Типът е едно от следните: “**sbyte**”, “**int**” or “**long**”.
* На втори ред ще получите **n** – брой на **ID-та**. **Всяко ID** ще бъде на отделен ред.

### Изход

Изведете **id** на крадеца.

### Ограничения

* Типа винаги ще бъде някое от следните: “**sbyte**”, “**int**” or “**long**”
* Интервалът за **sbyte** ще бъде **[-128…127]**
* **n** ще бъде в интервала **[1…20]**.
* Всички **id-та** ще бъдат цели числа в интервала **[-9223372036854775808…9223372036854775807]**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **sbyte**  **5**  1  126  128  1000  1241 | 126 |  | **long**  **4**  1  6  3  2 | 6 |

## \* Осъди крадеца

В предната задача хванахме крадеца, сега обаче трябва и да изчислим неговата присъда.

Неговата присъда е равна на броя на пътите, в които неговото ID препълва sbyte. Закръглете нагоре годините към най-близката **цяла стойност** (5.01 🡺 6).

Пример: Ако id-то е **5251**, това значи, че присъдата ще бъде равна на: **5251 / 127 = 41.35** години. Закръглено това са **42 години**.

Забележете, че **id-то** може да бъде отрицателно и да препълни отрицателната граница на **sbyte**.

### Вход

* На първи ред, ще получите типа на id **на крадеца**. Типа **винаги** ще бъде едно от следните: “**sbyte**”, “**int**” или “**long**”.
* На втория ред ще получите **n** – брой на **ид-та**. Всяко ще бъде на нов ред

### Изход

Ако **годините** на присъдата са повече от **1** изведете:

Prisoner with id {id of the thief} is sentenced to {duration of the sentence} years

В противен случай изведете:

Prisoner with id {id of the thief} is sentenced to {duration of the sentence} year

### Ограничения

* Типът **винаги** ще е едно от следните: “**sbyte**”, “**int**” или “**long**”
* За **sbyte** интервала е **[-128…127]**
* **n** ще е в интервала **[1…20]**
* **id-тата** ще са в интервала **[-9223372036854775808…9223372036854775807]**
* Няма да има **id** което да е **0**.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| **int**  **4**  -2561  -3412  -5999  -2641 | Prisoner with id -2561 is sentenced to 21 years | **-2561 / -128 = 20.01**. След закръгляне става **21 години**. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **sbyte**  **5**  1  126  128  1000  1241 | Prisoner with id 126 is sentenced to 1 year |  | **long**  **5**  1  56  100  -42  -2411 | Prisoner with id 100 is sentenced to 1 year |

## Строител

Вие сте строител и трябва да купите материали за един от вашите клиенти. Това е специална къща и ще има нужда от специални материали. Тази къща има нужда от **4** sbyte променливи и **10** int променливи. Груба скица на къщата изглежда по следния начин:



Ще получите две числа от конзолата, които ще бъдат цената на материалите. **Първото** ще бъде цяло число, а другото ще бъде sbyte, но няма да знаете в какъв ред ще бъдат дадени. Числото в intще бъде цената на int **материалите**, а числото в sbyteще е цената на sbyte **материалите**.

Изчислете **общата цена на материалите** и ги изведете на конзолата.

### Вход

* Ще получите два реда, съдържащи различни числа от различни типове.

### Изход

Изведете общата цена на материалите.

### Ограничения

* За sbyte: **[0…127]**
* За int**:** **[128…2147483647]**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| 100  2000 | 20400 |  | 2147483647  127 | 21474836978 |

## Създаване на дума

Напишете програма, която комбинира **n** знака и извежда на един ред **комбинацията** им

### Вход

* На **първи ред**, ще получите **n** – броят на **редовете**, които ще **следват**
* На следващите **n реда** – ще получите **малки** и **големи** букви от **английската** азбука

### Изход

**Изведете** на екрана **думата** във формата:

The word is: {word}

### Ограничения

* **n** е в интервала **[1…20]**.
* Знаците винаги ще бъдат **букви от английската азбука**
* Ще получавате по една буква на ред

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **5**  A  b  C  d  E | The word is: AbCdE |  | **9**  C  o  d  e  R  u  l  z  z | The word is: CodeRulzz |

## Сума на знаци

Напишете програма, която сумира ASCII кодовете на **n** знака и извежда сумата им.

### Вход

* На **първи ред**, ще получите **n** – броят на **редовете**, които ще **следват**
* На следващите **n реда** – ще получите буквите от латинската азбука

### Изход

Изведете общата сума в следния формат:

The sum equals: {totalSum}

### Ограничения

* **n** е в интервала **[1…20]**.
* Знаците винаги ще бъдат малки или големи латински букви
* Ще се въвежда по една буква на ред

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **5**  A  b  C  d  E | The sum equals: 399 |  | **12**  S  o  f  t  U  N  i  R  u  l  z  z | The sum equals: 1263 |

## Слепване на низове

Напишете програма, която въвежда **три** реда от конзолата. На **първи ред** ще въведете **разделител** (**char**) – трябва да **разделите** всички низове с този разделител. На **втори ред** ще получите “**even**” или “**odd**”. Ако получите “**odd**”, трябва да вземете всеки нечетен низ и обратното – ако получите “**even**”. На последния ред ще получите брой на редовете – **n**. Първата стъпка от цикъла започва от **1**.

Изведете получения низ на нов ред.

### Ограничения

* **n** ще бъде в интервала **[1…20]**.
* Низовете ще бъдат не по-дълги от **30** знака

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **изход** |  | **Вход** | **Изход** |
| -  **even**  **5**  One  Two  Three  Four  Five | Two-Four |  | **&**  **odd**  **4**  Pesho  Stefan  Maria  Gergana | Pesho&Maria |

### Подсказки

* В C#, може да ползвате [String.Remove(…)](https://msdn.microsoft.com/en-us/library/d8d7z2kk(v=vs.110).aspx) за да премахнете последния разделител

## Колички

Напишете програма, която изчислява обема на **n** бурета. Ще въведете общо **3 \* n** реда. **Всеки три реда ще съдържат информация за едно буре**. Първият ред е **модела** на бурето, втория е **радиуса** му, а третия е **височината** му.

Изчислете обема използвайки формулата: π \* r^2 \* h.

Накрая, изведете **модела** на най-голямото буре.

### Вход

Ще получите **3 \* n** реда. Всяка следваща група редове ще бъде на отделно:

* Първи – **model** – **string**.
* Втори –**radius** – **число с плаваща запетая** число
* Third – **height** – **integer** число

### Изход

Изведете **модела** на най-голямото буре.

### Ограничения

* **n** ще бъде в интервал **[1…10]**
* **радиусът** ще бъде **число с плаваща запетая** в интервала **[1…3.402823E+38]**
* **височината** ще бъде **цяло число** в интервала **[1…2147483647]**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| **3**  Keg 1  10  10  Keg 2  20  20  Keg 3  10  30 | Keg 2 |  | **2**  Smaller Keg  2.41  10  Bigger Keg  5.12  20 | Bigger Keg |

## Декриптиране на съобщение

Ще получите **ключ** (**цяло число**) и **n** знака след това. Добавете ключа към всеки то знаците и ги слепете към съобщението. Накрая изведете полученото съобщение.

### Вход

* На **първи ред**, ще получите **ключа**
* На **втори ред**, ще получите **n** – броя на редовете, които ще последват
* На следващите **n реда** – ще получите **малки** и **големи** букви от латинската азбука

### Изход

Изведете декриптираното съобщение.

### Вход

* **Ключът** ще бъде в интервала **[0…20]**
* **n** ще бъде в интервала **[1…20]**
* **Знаците** винаги ще бъдат малки или големи букви от латинската азбука
* Ще получавате по една буква на ред

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| **1**  **7**  C  d  b  q  x  o  s | Decrypt |

## \* Симулатор на лодка

Имате задача да направите симулатор на състезание с лодки. Ще получите две букви, които ще обозначават двете лодки.

След това ще получите **n случайни** низа. Всеки низ на нечетен ред показва скоростта на първата лодка, а на четен ред – скоростта на втората лодка. Лодката се придвижва с толкова позиции, колкото е дължината на съответния низ. Лодката, която първа стигне до **50**-та позиция е победител.

Лодките могат да се **ъпгрейдват**, което ще рече, че когато получим низа “**UPGRADE**” ще добавим 3 към **ASCII** кодовете и на двата знака на лодките и след това получените знаци ще се използват за визуализиране на лодките. Ако получите “**UPGRADE**”, лодките **НЕ се** мърдат.

Ако една от лодките стигне до **50** – изведете знака на победителя и спрете да приемате входни данни. Ако нито една от лодките не стигне до 50 – изведете тази, която е успяла да стигне до най-голямата позиция.

### Вход

* На **първи ред**, ще получите знака на първата лодка
* На **втори ред**, ще получите знака на втората лодка
* На **трети ред,** ще получите **n** – броя на редовете, които ще последват

### Изход

Изведете само знака на печелившата лодка.

### Ограничения

* **n** ще е в интервала **[1…20]**
* Дължината на низовете ще е между **[1…100]** знака
* В края**,** лодките няма да имат еднакви позиции

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| **!**  **(**  **7**  move  need for speed  go  fast and furious  UPGRADE  stopTheBoat  UPGRADE | . | Първа лодка🡺 ‘!’, втора лодка 🡺 ‘(’  “**move**” 🡺 **4 знака** 🡺 **първа лодка (нечетен ред)** премества се с **4 позиции**  “**need for speed**” 🡺 14 знака 🡺 **втора лодка (четен ред)** премества се с **14 позиции**  “**go**” 🡺 2 знака 🡺 **първа лодка (нечетен ред)** премества се с **2** позиции.  “**fast and furious**” 🡺 16 знака 🡺 **втора лодка (четен ред)** премества се с **16** позиции.  “**UPGRADE**” 🡺 добавяме 3 към ‘!’ 🡺 става **‘**$**’**, добавяме 3 към **‘**(**‘** 🡺 става **‘**+**’**.  “**stopTheBoat**” 🡺 11 знака 🡺 **втора лодка (четен ред)** премества се **11** позиции.  “**UPGRADE**”🡺 добавяме 3 към ‘$’ 🡺 става **‘**'**’**, добавяме 3 към **‘**+**‘** 🡺 става **‘**.**’**.  **победител – втора лодка 🡺 41 позиция >** **6 позиция** 🡺 **втората лодка печели** |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| **E**  **A**  **10**  UPGRADE  start  driveWithTheSpeedOfLight  go  driveWithTheSpeedOfLightOrFaster  Should not be read  a  Should not be read  b  Should not be read | H | Започваме с **UPGRADE** и първата лодка е представена с ‘**H**’, а втората с ‘**D**’  След **5ти** ред **първата** лодка е стигнала до **50** позиция и **НЕ** трябва да се приема вход от другите редове. |

## Векове към наносекунди

Напишете програма, в която въвеждаме цяло число – брой **векове** и го преобразувайте в **години, дни, часове, минути, секунди, милисекунди, микросекунди, наносекунди.**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 | 1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes = 3155673600 seconds = 3155673600000 milliseconds = 3155673600000000 microseconds = 3155673600000000000 nanoseconds |
| 5 | 5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes = 15778454400 seconds = 15778454400000 milliseconds = 15778454400000000 microseconds = 15778454400000000000 nanoseconds |

### Подсказки

* Използвайте подходящ тип данни за всяко преобразуване. Внимавайте с препълванията!
* Нека една година да има 365.2422 дни.

## Гласна или цифра

Напишете програма, която проверява дали даден символ е **цифра**, **гласна** или друг символ.

### Примери

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| A | vowel |  | 9 | digit |  | g | other |

## Рефакторирайте Проверка за просто число

Получавате код, който проверява дали числата в даден интервал [2...N] са прости. За всяко число изведете "{number} -> {True or False}". Кодът, все пак не е добре написан. Вашата работа е да го подобрите, така че да е лесен за четене и разбиране**.**

### Код

|  |
| --- |
| **Примерен код** |
| int \_\_\_Do\_\_\_ = int.Parse(Console.ReadLine());  for (int DAVIDIM = 0; DAVIDIM <= \_\_\_Do\_\_\_; DAVIDIM++)  { bool TowaLIE = true;  for (int delio = 2; delio <= Math.Sqrt(DAVIDIM); delio++)  {  if (DAVIDIM % delio == 0)  {  TowaLIE = false;  break;  }  }  Console.WriteLine($"{DAVIDIM} -> {TowaLIE}");  } |

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5 | 2 -> True  3 -> True  4 -> False  5 -> True |

### Подсказки

* Проверете в Интернет как да разберете дали едно число е просто
* Преименувайте всички променливи така че да е ясно каква е тяхната роля в алгоритъма

## \* Сравняване на реални числа

Напишете програма, която да сравнява числа с плаваща запетая сигурно, като точността трябва да е eps = 0.000001. Забележете, че не можем директно да сравним две числа с плаваща запетая a и b чрез a==b заради природата на аритметиката на числата с плаваща запетая. Затова приемаме, че две числа са еднакви, ако те са по-близо едно до друго от зададен **eps**.

Ще получите **два** реда, всеки от тях съдържа **число с плаваща запетая**. Вашата задача е да сравните стойностите на двете числа.

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Число a** | **Число b** | **Равни (с точност eps=0.000001)** | **Обяснение** |
| 5.3 | 6.01 | False | Разликата от 0.71 е прекалено голяма (> eps) |
| 5.00000001 | 5.00000003 | True | Разликата 0.00000002 < eps |
| 5.00000005 | 5.00000001 | True | Разликата 0.00000004 < eps |
| -0.0000007 | 0.00000007 | True | Разликата 0.00000077 < eps |
| -4.999999 | -4.999998 | False | Граничен случай. Разликата 0.000001 == eps. Приемаме числата за различни. |
| 4.999999 | 4.999998 | False | Граничен случай. Разликата 0.000001 == eps. Приемаме числата за различни. |

## Изведете част от ASCII таблицата

Намерете в Интернет повече информация за [ASCII](http://www.ascii-code.com/) (American Standard Code for Information Interchange) и напишете програма, която извежда част от **ASCII таблицата** от знаци на конзолата. На първи ред на входа ще получите индекса на знака, от който трябва да започнете, а на втория ред – индекса на последния знак.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 60  65 | < = > ? @ A |
| 69  79 | E F G H I J K L M N O |
| 97  104 | a b c d e f g h |
| 40  55 | ( ) \* + , - . / 0 1 2 3 4 5 6 7 |

## \* Различни размери на цяло число

Дадено е цяло число. Трябва да определите в кой тип може да се впише числото.

### Вход

* Получавате **N** – цяло число, което може да бъде произволно голямо или малко

### Изход

Трябва да определите дали може да се запише в някой от примитивните типове. Ако може, изведете:

|  |
| --- |
| {N} can fit in:  \* dataType |

Ако има повече от един подходящ тип, изведете всеки на отделен ред, като ги подредите в следната подредба   
(sbyte **<** byte **<** short **<** ushort **<** int **<** uint **<** long).

Ако числото не може да се запише в един от четирите по-горе споменати типове, изведете:

|  |
| --- |
| {N} can't fit in any type |

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| -150 | -150 can fit in:  \* short  \* int  \* long | 150000 | 150000 can fit in:  \* int  \* uint  \* long |

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1500000000 | 1500000000 can fit in:  \* int  \* uint  \* long |

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 213333333333333333333333333333333333 | 213333333333333333333333333333333333 can't fit in any type |

### Подсказки

Използвайте **try … catch** конструкция.

## Министерство на образованието и науката (МОН)

* Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".



* Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен** **лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).

