# Допълнително упражнение: Цикли

Тествайте решенията си в **Judge системата**: <https://judge.softuni.bg/Contests/3158/Loops-More-Exercises>

## Числа до 1000, завършващи на 7

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

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| (няма) | 7  17  27  …  997 |

### Насоки

1. Направете for**-цикъл** от **7 до 997**
2. На **всяка итерация** на цикъла проверете дали числото **завършва на 7**. Едно число завършва на 7, когато резултатът от **модулното деление на числото и 10 е равен на 7**.

Text

Description automatically generated

## Число, кратно на 10

Напишете програма, която чете цяло **число** n, въведено от потребителя. При въвеждане на число, което **не се дели на 10 без остатък**, да се отпечата **съобщение за** **грешка** и потребителят да се подкани **да въведе ново число**, докато **не въведе** **подходящо число**.

### Примери

|  |
| --- |
| **Вход/Изход** |
| 40  The number is: 40 |
| **105**  Invalid number!  **101**  Invalid number!  **-202**  Invalid number!  -20  The number is: -20 |

### Насоки

1. Прочетете от конзолата **числото**, въведено от потребителя



1. Създайте **while-цикъл**, който итерира **докато числото не се дели на 10 без остатък**

**Chart

Description automatically generated with medium confidence**

1. На **всяка итерация** на цикъла отпечатайте **“Invalid number!”** и прочетете **ново число**

Text

Description automatically generated

1. **След цикъла** отпечатайте **валидното число**

## Сума на двойки

Напишете програма, която чете **двойка числа** и пресмята **сумата** на всяка от тях. Ако **сумата на една двойка** е **по-голяма** от дадено число **n**, изписва "Bigger Sum!" на конзолата.

От конзолата се четат:

* **n** – цяло число, с което ще се **сравнява сумата**
* **m** – цяло число в диапазона [1..100], което представлява **броят на двойките числа**, които ще се въведат на следващите редове
* На следващите **m-реда** – **по две цели числа** на отделни редове (**една двойка**)

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 1  2  3  2  0  0 | Bigger Sum! | Числото **n** е 1, тоест търсим сума, по-голяма от 1.  Числото **m** e 2, тоест ще имаме 2 двойки числа (4 реда).  Първа двойка – 3 + 2 = 5 => 5 > 1 => изписваме "**Bigger Sum!**"  Втора двойка – сборът им е 0 => не изписваме нищо |

## Сумиране на гласните букви

Да се напише програма, която чете **текст**, въведен от потребителя, изчислява и отпечатва **сумата от стойностите на гласните букви** според таблицата по-долу:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| буква | a | e | i | o | u |
| стойност | 1 | 2 | 3 | 4 | 5 |

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| hello | 6 | e + o = 2 + 4 = 6 |
| hi | 3 | i = 3 |
| bamboo | 9 | a + o + o = 1 + 4 + 4 = 9 |

## Събиране на думи

Напишете програма, която при **команда** "Go" събира по **три думи** и ги **принтира като една**. При команда "End" програмата приключва.

### Вход

От конзолата се четат:

* **Команда** – текст с възможности "**Go**" и "**End**"
* При команда "Go" се четат
  + **три реда** с по една дума
  + **нова команда** ("**Go**" или "**End**")

Програмата приключва при команда "End"

### Изход

Програмата принтира **всяка новообразувана дума** от подадените през конзолата 3 думи за събиране.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| Go  apple  lemon  berry  Go  how  are  you  End | applelemonberry  howareyou | Получаваме "Go", съответно четем три реда и събираме думите в една: apple + lemon + berry = **applelemonberry** и **принтираме** новата дума.  Отново получаваме "Go", четем думите и ги събираме в една.  Получаваме "End" и програмата приключва. |

## Умната Лили

Лили вече е на **N години**. За всеки свой **рожден ден** тя получава подарък. За **нечетните** рождени дни (**1, 3, 5...n**) получава **играчки**, а за **четните** (**2, 4, 6...n**) получава **пари**. За **втория рожден ден** получава **10.00 лв**., като сумата се **увеличава с 10.00 лв. за всеки следващ четен рожден ден** (2 -> 10, 4 -> 20, 6 -> 30...и т.н.). В годините, в които тя **получава пари**, брат ѝ **взима по 1.00 лв.** от тях. Лили **продала играчките**, получени през годините, **всяка за P лв.** и **добавила** сумата **към спестените пари**. С парите искала да си **купи пералня за X лева**. Напишете програма, която да пресмята **колко пари е събрала** и дали ѝ **стигат да си купи пералня**.

### Вход

Програмата прочита **3 числа**, въведени от потребителя, на отделни редове:

* **Възрастта** на Лили - цяло число в интервала [1...77]
* **Цената на пералнята** – реално число
* **Цената на една играчка** – реално число

### Изход

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

* Ако парите на Лили са **достатъчни**:
  + **“**Yes! {N}**”** - където **N** е остатъка пари след покупката
* Ако парите **не са достатъчни**:
  + **“No! {М}“** - където **M** е сумата, която не достига
* Числата **N** и **M** трябва да са форматирани **до втория знак след десетичната запетая**.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 10  170.00  6 | Yes! 5.00 | **Първи рожден ден** получава **играчка**; **2ри -> 10лв**; 3ти -> играчка;  **4ти** -> 10 + 10 = **20лв**; **5ти** -> играчка; **6ти** -> 20 + 10 = **30лв**; **7ми** -> играчка;  **8ми** -> 30 + 10 = **40лв**; **9ти** -> играчка; **10ти** -> 40 + 10 = **50лв**.  **Спестила е** -> 10 + 20 + 30 + 40 + 50 = **150лв**. Продала е **5 играчки по 6 лв.** = **30лв**.  **Брат ѝ взел 5 пъти по 1 лев** = **5лв**. **Остават** -> 150 + 30 – 5 = **175лв**.  **175 >= 170** (цената на пералнята) **успяла** е да я купи и са и **останали** 175-170 = **5 лв.** |
| 21  1570.98  3 | No! 997.98 | **Спестила е 550лв**. **Продала** е **11** играчки по **3 лв.** = **33лв**. Брат ѝ **взимал 10** години по **1 лев** = **10лв**. **Останали** 550 + 33 – 10 = **573лв**  **573 < 1570.98** – **не е успяла** да купи пералня. **Не ѝ достигат** 1570.98–573 = **997.98лв** |

## Дартс

Вашата задача е да напишете програма, която да изчислява **дали даден играч е успял да спечели лег**. (Лег се нарича единична игра на дартс)

Първоначално играчът **започва с 301 точки**. Играчът хвърля стрелата върху таблото, като за всяко улучено поле, той получава **определен брой точки**. Всяко поле има по **три сектора**: **единичен** (Single) сектор, от който се взима броят точки от полето. **Двоен** (Double), от него се взимат **удвоените** **точки** от полето и **троен** (Triple) сектор, точките от който са **умножени по 3**.

Получените точки от всеки изстрел се **изваждат от началните точки**, до достигане на **0**.

**Забележка:** При изстрел, даващ **повече точки от наличните**, той се зачита за **неуспешен** и играчът трябва да хвърля **отново**, докато не уцели точки равни на оставащите или по-малки - такъв удар се счита за успешен.

**Пример:** При наличниточки 100, удар даващ повече от 100 точки е неуспешен. При налични точки 100, удар даващ по-малко или равни на 100 точки е успешен.

### Вход

Първоначално се чете **един ред**:

* **Името на играча** - текст

След това до получаване на **команда "Retire"** се четат **многократно по два реда:**

* **Поле –** текст- **"**Single**"**, **"**Double**"** или **"**Triple**"**
* **Точки** – цяло число в интервала [0… 100]

### Изход

Играта приключва при въвеждане на команда "Retire" или при изравняване на началните 301 точки към 0. На конзолата трябва да се отпечата **един ред**:

* Ако играчът **е спечелил лега:**
  + **"{името на играча} won the leg with {успешните изстрели} shots."**
* Ако играчът **се е отказал от играта**:
  + **"{името на играча} retired after {неуспешни изстрели} unsuccessful shots."**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| Michael van Gerwen  Triple  20  Triple  19  Double  10  Single  3  Single  1  Triple  20  Triple  20  Double  20 | Michael van Gerwen won the leg with 8 shots. | Започваме със 301 точки  Първият удар е тройно 20 -> 60 <= 301  301 – 60 = 241; успешни изстрели = 1  Втори удар е тройно 19 -> 57 <= 241  241 – 57 = 184; успешни изстрели = 2  Трети удар е двойно 10 -> 20 <=184  184 – 20 = 164; успешни изстрели = 3  Четвърти удар е единично 3 -> 3 <= 164  164 – 3 = 161; успешни изстрели = 4  Пети удар е единично 1 -> 1 <= 161  161 – 1 = 160; успешни изстрели = 5  Шести удар е тройно 20 -> 60 <= 160  160 – 60 = 100; успешни изстрели = 6  Седми удар е тройно 20 -> 60 <= 100  100 – 60 = 40; успешни изстрели = 7  Осми удар е двойно 20 -> 40 <=40  40 – 40 = 0; успешни изстрели = 8 |
| Stephen Bunting  Triple  20  Triple  20  Triple  20  Triple  20  Triple  20  Triple  20  Double  7  Single  12  Double  1  Single  1 | Stephen Bunting won the leg with 6 shots. | Започваме със 301 точки  Първият удар е тройно 20 -> 60 <= 301  301 – 60 = 241; успешни изстрели = 1  .  .  .  Петият удар е тройно 20 -> 60 <= 61  61 – 60 = 1; успешни изстрели = 5  Шестият удар е тройно 20 -> 60 > 1  Неуспешни изстрели = 1  Седмият удар е двойно 7 -> 14 > 1  Неуспешни изстрели = 2  Осмият удар е единично 12 -> 12 > 1  Неуспешни изстрели = 3  Деветият удар е двойно 1 -> 2 > 1  Неуспешни изстрели = 4  Десетият удар е единично 1 -> 1 <= 1  1 – 1 = 0; успешни изстрели = 6 |
| Rob Cross  Triple  20  Triple  20  Triple  20  Triple  20  Double  20  Triple  20  Double  5  Triple  10  Double  6  Retire | Rob Cross retired after 3 unsuccessful shots. | Започваме със 301 точки  Първият удар е тройно 20 -> 60 <= 301  301 – 60 = 241; успешни изстрели = 1  …  Пети удар е двойно 20 -> 40 <= 61  61 – 40 = 21; успешни изстрели = 5  Шести удар е тройно 20 -> 60 > 21  Неуспешни изстрели 1  Седми удар е двойно 5 -> 10 <= 21  21 – 10 = 11; успешни изстрели 6  Осми удар е тройно 10 -> 30 > 11  Неуспешни изстрели 2  Девети удар е двойно 6 -> 12 > 11  Неуспешни изстрели 3  Retire -> играча се отказва, след 3 неуспешни изстрела |

## Помогни на Димо

Димо е професионален лекоатлет. За да следи **напредъка си** и дали е **постигнал** **нужните резултати**, той се нуждае от програма. Напишете програмата, за да помогнете на Димо. Първо се прочита **дължината на скока**, който Димо трябва да изпълни – тя е неговата **цел**. Първоначално, той опитва да изпълни **скок с дължина 30см по-малка от целта**. Димо може да направи само **три опита** за скок за **всяка моментна целева дължина**, като опитът му е **успешен**, в случай че **дължината на скока на опита е по-голяма от дължината на моментната цел**. Когато лекоатлетът **успее да скочи моментната целева дължина**, тя се **увеличава с 5 сантиметра**. Ако Димо **не успее** да скочи над моментната целева дължина **3 пъти**, той вече няма опити и тренировката му се счита за **неуспешна**. Ако се **достигне целевата дължина** и Димо направи скок над тази дължина, тренировката е **успешна**.

### Вход

На входа последователно се подават цели числа [100…300]:

* На първия ред – **целевата дължина** в сантиметри
* До приключване на програмата, на следващите редове една по една се прочитат **дължините, на които Димо е успял да скочи всеки път**

### Изход

Отпечатайте **един ред** на конзолата:

* Ако Димо **не** **скочи над целевата дължина**:
  + **"{моментната целева дължина}cm was too hard for Dimo to reach. He made {общия брой опити} tries."**
* Ако Димо **скочи над целевата дължина:**
  + **"Dimo did it, he reached his goal with {височината на последната моментната целева дължина }cm. He made {общия брой опити} tries."**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 221  195  202  203  218  219  220  225 | Dimo did it, he reached his goal with 221cm. He made 7 tries. | Желана дължина: 221 -> Димо започва от **191**  Първи път, моментната целева дължина 191, 195 е успешен скок  Увеличаване на моментната целева дължина с 5см: 191->196.  Втори път, моментна целева дължина 196, 202 е успешен скок  Увеличаване на моментната целева дължина с 5см.  Трети път, височина на летвата 201, 203 е успешен скок  …  Седми път, моментна целева дължина 221, целева дължина 225, 225 е успешен скок -> Димо успява. |
| **Вход** | **Изход** | **Обяснения** |
| 240  215  214  215  218  221  225  224  225 | 225cm was too hard for Dimo to reach. He made 8 tries. | Желана дължина: 240 -> Димо започва от 210  Първи път, моментна целева дължина: 210, 215 е успешен скок  Увеличаване на моментната целева дължина с 5см: 210 -> 215.  Втори път, моментна целева дължина 215, 214 е неуспешен скок  Трети път и втори опит на моментна целева дължина 215, 215 е неуспешен скок  Четвърти път и трети опит на 215, 218 е успешен скок  Увеличаване на моментната целева дължина с 5см.  Пети път, моментна целева дължина 220, 221 е успешен скок  Увеличаване на моментната целева дължина с 5см.  Шести път, моментна целева дължина 225, 225 е неуспешен скок  Седми път и втори опит на 225, 224 е неуспешен скок  Осми път и трети (последен) опит на 225, 225 е неуспешен опит -> Димо не успява да прескочи целта. |

## Комбинации

Напишете програма, която изчислява **колко решения в естествените числа** (включително и нулата) има уравнението:

**x1 + x2 + x3 = n**

**Числото n е цяло число** и се въвежда от конзолата.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 25 | 351 | Генерираме всички комбинации от 3 числа, като първата е:  0+0+0=0, но понеже не е равна на 25, продължаваме:  0+0+1=1 – също не е 25 и т.н.  Стигаме до първата валидна комбинация:  0 + 0 + 25 = 25, увеличаваме броя на валидни комбинации с 1,втората валидна комбинация е:  0 + 1 + 24 = 25  Третата:  0 + 2 + 23 = 25 и т.н.  След генериране на всички възможни комбинации, броят на валидните е 351. |
| 20 | 231 |  |
| 5 | 21 |  |

## Пътуване

Ани обича да пътува и иска тази година да посети няколко различни дестинации. Като си избере дестинация, ще прецени **колко пари ще ѝ трябват**, за да отиде до там, и ще започне да **спестява**. Когато е спестила **достатъчно**, ще може да пътува.

От конзолата всеки път ще се четат **първо дестинацията** и **минималния бюджет**, който ще е нужен за пътуването.

След това ще се четат **няколко суми**, които Ани **спестява** като работи и когато успее да **събере достатъчно за пътуването**, ще заминава, като на конзолата трябва да се изпише:

**"Going to {дестинацията}!"**

Когато е посетила всички дестинации, които иска, **вместо дестинация ще въведе "End"** и програмата ще приключи.

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| Greece  1000  200  200  300  100  150  240  Spain  1200  300  500  193  423  End | Going to Greece!  Going to Spain! | France  2000  300  300  200  400  190  258  360  Portugal  1450  400  400  200  300  300  Egypt  1900  1000  280  300  500  End | Going to France!  Going to Portugal!  Going to Egypt! |

## Билети за кино

Вашата задача е да напишете програма, която да изчислява **процента на билетите за всеки тип от продадените билети**: **студентски** (**student**), **стандартен** (**standard**) и **детски** (**kid**) за всички прожекции. Трябва да изчислите и **колко процента от залата е запълнена** за всяка една прожекция.

### Вход

Входът е поредица от **цели числа** и **текст**:

* На първия ред до получаване на **командата** "**Finish**" - **име на филма** – текст
* На втори ред – **свободните места** в салона за всяка прожекция – цяло число [1 … 100]
* За всеки филм, се чете по **един ред** до изчерпване на свободните места в залата или до получаване на командата "**End**":
  + **Типът на закупения билет** - текст ("**student", "standard", "kid"**)

### Изход

На конзолата трябва да се отпечатат **следните редове**:

* След всеки филм да се отпечата, **колко процента от кино залата е пълна**

**"{името на филма} - {процент запълненост на залата}% full."**

* При получаване на **командата "Finish"** да се отпечатат **четири реда**:
  + **"Total tickets: {общият брой закупени билети за всички филми}"**
  + **"{процент на студентските билети}% student tickets."**
  + **"{процент на стандартните билети}% standard tickets."**
  + **"{процент на детските билети}% kids tickets."**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 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. | Първи филм – Taxi, местата в залата са 10  Купуват се 3 стандарти, 2 студентски, 1 детски билет и получаваме командата End.  Общо 6 билета от 10 места -> 60% от залата е заета.  Втори филм – Scary Movie, места в залата са 6  Купуват се 6 студентски билета и местата в залата свършват.  Общо 6 билета от 6 места -> 100% от залата е заета.  Получаваме командата Finish  Общо закупените билети за всички филми са 12.  За всички филми са закупени общо:  8 студентски билета. 8 билета от общо 12 е 66.67%  3 стандартни билета. 3 билета от общо 12 е 25%  1 детски билет. 1 билет от общо 12 е 8.33% |
| **Вход** | **Изход** | **Обяснения** |
| 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. | Първи филм – The Matrix, местата в залата са 20  Купуват се 2 стандартни, 4 студентски, 2 детски билета и получаваме командата End.  Общо 8 билета от 20 места -> 41.18% от залата е заета  Втори филм - The Green Mile, местата в залата са 17  Купуват се 3 стандартни, 3 студентски билета и получаваме командата End.  Общо 6 билета от 17 места -> 47.06% от залата е заета  Трети филм – Amadeus, местата в залата са 3  Купуват се 3 стандартни билета и местата в залата свършват.  Общо 3 билета от 3 места -> 100% от залата е заета.  Получаваме командата Finish  Общо закупените билети за всички филми са 17.  За всички филми са закупени общо:  7 студентски билета. 7 билета от общо 17 е 41.18%  8 стандартни билета. 8 билета от общо 17 е 47.06%  2 детски билета. 2 билета от общо 17 е 11.76% |

## Елемент, равен на сумата на останалите

Да се напише програма, която чете n на брой цели числа, въведени от потребителя, и проверява дали сред тях съществува число, което е равно на сумата на всички останали. Ако има такъв елемент, отпечатва "Yes", "Sum = " + **неговата стойност**; иначе отпечатва "No", "Diff = " + **разликата между най-големия елемент и сумата на останалите** (по абсолютна стойност).

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 7  3  4  1  1  2  **12**  1 | Yes  Sum = 12 | 3 + 4 + 1 + 2 + 1 + 1 = 12 |
| 4  **6**  1  2  3 | Yes  Sum = 6 | 1 + 2 + 3 = 12 |
| 3  1  1  10 | No  Diff = 8 | |10 - (1 + 1)| = 8 |
| 3  5  5  1 | No  Diff = 1 | |5 - (5 + 1)| = 1 |
| 3  1  1  1 | No  Diff = 1 |  |

## Четни / нечетни позиции

Напишете програма, която чете n на брой **числа**, въведени от потребителя, и пресмята **сумата**, **минимума** и **максимума** на числата на **четни** и **нечетни** позиции (броим от 1). Когато **няма** минимален / максимален елемент, отпечатайте "No".

**Изходът** да се форматира в следния вид:

"OddSum=" + {**сума** на числата на **нечетни** позиции},

"OddMin=" + { **минимална** стойност на числата на **нечетни** позиции } / {“No”},

"OddMax=" + { **максимална** стойност на числата на **нечетни** позиции } / {“No”},

"EvenSum=" + { **сума** на числата на **четни** позиции },

"EvenMin=" + { **минимална** стойност на числата на **четни** позиции } / {“No”},

"EvenMax=" + { **максимална** стойност на числата на **четни** позиции } / {“No”}

Всяко число трябва да е форматирано **до втория знак след десетичната запетая.**

### Примери

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| 6  **2**  3  **5**  4  **2**  1 | OddSum=9.00, OddMin=2.00, OddMax=5.00, EvenSum=8.00, EvenMin=1.00, EvenMax=4.00 | 2  **1.5**  -2.5 | OddSum=1.50, OddMin=1.50, OddMax=1.50, EvenSum=-2.50, EvenMin=-2.50, EvenMax=-2.50 | 1  **1** | OddSum=1.00, OddMin=1.00, OddMax=1.00, EvenSum=0.00, EvenMin=No, EvenMax=No | 0 | OddSum=0.00, OddMin=No, OddMax=No, EvenSum=0.00, EvenMin=No, EvenMax=No |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| 5  **3**  -2  **8**  11  **-3** | OddSum=8.00, OddMin=-3.00, OddMax=8.00, EvenSum=9.00, EvenMin=-2.00, EvenMax=11.00 |  | 4  1.5  **1.75**  1.5  **1.75** | OddSum=3.00, OddMin=1.50, OddMax=1.50, EvenSum=3.50, EvenMin=1.75, EvenMax=1.75 |  | 1  **-5** | OddSum=-5.00, OddMin=-5.00, OddMax=-5.00, EvenSum=0.00, EvenMin=No, EvenMax=No |  | 3  **-1**  -2  **-3** | OddSum=-4.00, OddMin=-3.00, OddMax=-1.00, EvenSum=-2.00, EvenMin=-2.00, EvenMax=-2.00 |

## 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%.

### Примери

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| **3**  1  2  999 | 66.67%  0.00%  0.00%  0.00%  33.33% | **4**  53  7  56  999 | 75.00%  0.00%  0.00%  0.00%  25.00% | **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 nнаброй **цели числа** в интервала [1…1000]. От тях някакъв процентp1 **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%.

* Процентът на числата, които **се делят на 2**
* Процентът на числата, които **се делят на** **3**
* Процентът на числата, които **се делят на 4**

### Примери

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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% |

## Старата Библиотека

Ани отива до родния си град след много дълъг период извън страната. Прибирайки се вкъщи, тя вижда старата библиотека на баба си и си спомня за любимата си книга. Помогнете на Ани, като напишете програма, в която тя въвежда търсената от нея **книга** (**текст**). **Докато Ани не намери любимата си книга** или **не провери всички в библиотеката**, програмата трябва да чете всеки път на нов ред **името на всяка следваща книга (текст).** Книгите в библиотеката са свършили щом получите **команда "No More Books".**

* Ако **не открие търсената книгата** да се отпечата на два реда**:**
* **"The book you search is not here!"**
* **"You checked {брой} books."**
* Ако **открие книгата си** се отпечатва един ред**:**
  + **"You checked {брой} books and found it."**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| Troy  Stronger  Life Style  Troy | You checked 2 books and found it. | Книгата която Ани търси, в случая е Troy. Първата книга от библиотеката е Stronger, втората е Life Style, третата книга е търсената – Troy и програмата приключва. |
| The Spot  Hunger Games  Harry Potter  Torronto  Spotify  No More Books | The book you search is not here!  You checked 4 books. | Книгата, която търси Ани е "The Spot".  Първата книга от библиотеката е Hunger Games, втората Harry Potter, третата Torronto, а четвъртата Spotify. Понеже няма повече книги в библиотеката, четенето на имена приключва. Ани не намери книгата, която търсеше. |
| Bourne  True Story  Forever  More Space  The Girl  Spaceship  Strongest  Profit  Tripple  Stella  The Matrix  Bourne | You checked 10 books and found it. |  |

## Подготовка за изпит

Напишете програма, в която Марин решава задачи от изпити **докато** **не** **получи** съобщение **"Enough"** отлектораси. При всяка решена задача той получава оценка. Програмата трябва да **приключи** прочитането на данни при **команда "Enough"** или акоМарин **получи определения брой незадоволителни оценки.**  
**Незадоволителна** е всяка оценка, която е **по-малка или равна на 4.**

### Вход

* **На първи ред - брой незадоволителни оценки** - цяло число в интервала [1…5]
* След това **многократно** се четат **по два реда:**
  + **Име на задача -** текст (низ)
* **Оценка -** цяло число в интервала [2…6]

### Изход

* Ако Марин стигне до командата **"Enough", отпечатайте на 3 реда:**
* **"Average score: {средна оценка}"**
* **"Number of problems: {броя на всички задачи}"**
  + **"Last problem: {името на последната задача}"**
* Ако получи **определеният брой незадоволителни оценки:**
* **"You need a break, {брой незадоволителни оценки} poor grades."**

**Средната оценка** да бъде форматирана **до втория знак след десетичната запетая.**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 3  Money  6  Story  4  Spring Time  5  Bus  6  Enough | Average score: 5.25  Number of problems: 4  Last problem: Bus | Броя на позволени незадоволителни оценки е 3.  Първата задача се казва Money, оценката на Марин е 6.  Втората задача е Story, оценката на Марин е 4.  Третата задача е Spring Time, оценката на Марин е 5.  Четвъртата задача е Bus, оценката на Марин е 6.  Следващата команда е Enough, програмата приключва.  Средна оценка: 21 / 4 = 5.25  Брой решени задачи: 4  Последна задача: Bus |
| **Вход** | **Изход** | **Обяснения** |
| 2  Income  3  Game Info  6  Best Player  4 | You need a break, 2 poor grades. | Броят незадоволителни оценки е 2.  Първата задача е Income, оценката на Марин е 3.  Втората задача е Game Info, оценката на Марин е 6.  Третата задача е Best Player, оценката на Марин е 4.  Марин достигна допустимия брой незадоволителни оценки, време е за почивка. |

## Почивка

Джеси е решила да събира пари за екскурзия и иска от вас да ѝ помогнете да разбере **дали ще успее да събере необходимата сума**. Тя **спестява** или **харчи част от** **парите** си всеки ден. Ако иска да **похарчи повече от наличните си пари**, то тя ще похарчи **колкото има и ще ѝ останат 0 лева**.

### Вход

От конзолата се четат:

* **Пари нужни за екскурзията** - реално число в интервала [1.00.. .25000.00]
* **Налични пари** - реално число в интервала [0.00... 25000.00]

След това **многократно** се четат **по два реда:**

* **Вид действие** – текст с възможности "spend" и "save".
  + **Сумата, която ще спести/похарчи** - реално число в интервала [0.01… 25000.00]

### Изход

Програмата трябва да приключи при следните случаи:

* Ако **5 последователни дни** Джеси **само харчи,** на конзолата да се изпише**:**
* **"You can't save the money."**
* **"{Общ брой изминали дни}"**
* Ако Джеси **събере парите за почивката** на конзолата се изписва:
* **"You saved the money for {общ брой изминали дни} days."**

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 2000  1000  spend  1200  save  2000 | You saved the money for 2 days. | Пари нужни за почивката: 2000  Наличните пари: 1000  spend – изваждаме от парите следващото число  1200 - ние разполагаме с 1000, но се опитваме да похарчим 1200, тъй като не разполагаме с толкова, харчим наличните си 1000 и оставаме 0 лева.  save – прибавяме към парите следващото число  2000 – разполагаме с 0, добавяме 2000 и събираме парите успешно за 2 дни. | |
| 110  60  spend  10  spend  10  spend  10  spend  10  spend  10 | You can't save the money.  5 | 250  150  spend  50  spend  50  save  100  save  100 | You saved the money for 4 days. |

## Стъпки

Габи иска да започне здравословен начин на живот и си е поставила за цел да върви **10 000 стъпки всеки ден**. Някои дни обаче е много уморена от работа и ще иска да се прибере преди да постигне целта си. Напишете програма, която чете от конзолата **по колко стъпки изминава** тя всеки път като излиза през деня и **когато постигне целта си**,изписва "**Goal reached! Good job!**" и колко стъпки повече е извървяла "{разликата между стъпките} steps over the goal!"

Ако иска да се **прибере преди това**, тя ще въведе **командата** "Going home" и **ще въведе стъпките**, които е извървяла, докато се прибира. След това, ако **не е успяла** да постигне целта си, на конзолата трябва да се изпише: "{разликата между стъпките} more steps to reach goal."

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 1000  1500  2000  6500 | Goal reached! Good job!  1000 steps over the goal! | 1500  300  2500  3000  Going home  200 | 2500 more steps to reach goal. |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 1500  3000  250  1548  2000  Going home  2000 | Goal reached! Good job!  298 steps over the goal! | 125  250  4000  30  2678  4682 | Goal reached! Good job!  1765 steps over the goal! |

## Монети

Производителите на вендинг машини искали да направят машините си да връщат възможно **най-малко монети ресто**. Напишете програма, която приема **сума** - **рестото**, което трябва да се върне, и изчислява **с колко най-малко монети може да стане това**.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 1.23 | 4 | Рестото ни е 1 лев и 23 стотинки. Машината ни го връща с 4 монети: монета от 1 лев, монета от 20 стотинки, монета от 2 стотинки и монета от 1 стотинка. |
| 2 | 1 | Рестото ни е 2 лева. Машината ни го връща с 1 монета от 2 лева. |
| 0.56 | 3 | Рестото ни е 56 стотинки. Машината ни го връща с 3 монети: монета от 50 стотинки, монета от 5 стотинки и монета от 1 стотинка. |
| 2.73 | 5 | Рестото ни е 2 лева и 73 стотинки. Машината ни го връща с 5 монети: монета от 2 лева, монета от 50 стотинки, монета от 20 стотинки, монета от 2 стотинки и монета от 1 стотинка. |

## Торта

Поканени сте на 30-ти рожден ден, на който рожденикът черпи с огромна торта. Той обаче не знае **колко парчета могат да си вземат гостите от нея**. Вашата задача е да напишете програма, която изчислява **броя на парчетата**, които гостите са взели, преди тя да **свърши**. Ще получите **размерите на тортата** (широчина и дължина – цели числа в интервала [1...1000]), и след това на всеки ред, до получаване на **командата** "STOP" или **докато не свърши тортата**, **броят на парчетата**, които гостите вземат от нея.

**Бележка:** Едно парче торта е с размер 1х1 см.

Да се **отпечата** на конзолата **един** от следните редове:

* "{брой парчета} pieces are left." - ако стигнете до STOP и не са свършили парчетата торта
* **"No more cake left! You need {брой недостигащи парчета} pieces more."**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 10  10  20  20  20  20  21 | No more cake left! You need 1 pieces more. | Размер на тортата: 10\*10 = 100.  Въвеждат се многократно брой парчета които са взети:  20+20+20+20+21=101  Не ни достига едно парче: 101-100=1 |
| 10  2  2  4  6  STOP | 1. pieces are left. |  | |

## Пирамида от числа

Напишете програма, която чете цяло число n, въведено от потребителя, и отпечатва **пирамида от числа** като в примерите:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| 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 |

## GUI App – Пирамида от числа

Реализирайте предходната задача като **Windows Forms GUI app**. Потребителският интерфейс трябва да изглежда така:

Graphical user interface

Description automatically generated

### Насоки

1. Създайте нов **Windows Forms проект** и му задайте **смислено име**, например "**NumberPyramidApp**"
2. Променете **името** на формата: "**Form1**" 🡪 "**FormNumberPyramid**"
3. Променете **заглавието** на формата: "**Form 1**" 🡪 "**Number Pyramid**"
4. Добавете необходимите **контроли**, примерно ето така:

Graphical user interface, application, Word

Description automatically generated

Контролите трябва да са следните:

* **два** надписа (**Label**):
  + "**labelMaxNum**" с текст "**Max number:** "
  + "**labelPyramid**" с текст "**Result**: "
* **едно** числово поле (**NumericUpDown**):
  + "**numericUpDownMaxNum**" с максимална стойност **20**
* **един** бутон (**Button**):
  + "**buttonGeneratePyramid**" с текст "**Generate pyramid**"

1. Добавете **метод-обработчик** на бутона **buttonDisplayRooms** при събитието **Click** (чрез двоен клик в празното пространство)
2. Добавете код в метода-обработчик, който да визуализира пирамидата в **labelPyramid**
3. Стартирайте приложението с **[Ctrl + F5]** и го тествайте:

Graphical user interface

Description automatically generated Graphical user interface, application

Description automatically generated

## Суми прости и непрости числа

Напишете програма, която чете от конзолата цели числа в диапазона от **-**2,147,483,648 до 2,147,483,647, докато не се получи **команда** "**stop"**. Да се намери **сумата** на всички въведени **прости** и **сумата** на всички въведени **непрости** числа. Тъй като по дефиниция от математиката **отрицателните числа не могат да бъдат прости**, ако на входа се подаде **отрицателно** число, трябва да се изведе следното съобщение "**Number is negative.**". В този случай въведено число се **игнорира** и не се прибавя към нито една от двете суми, а програмата **продължава своето изпълнение**, очаквайки въвеждане на следващо число.

На **изхода** да се отпечатат на два реда **двете намерени суми** в следния формат:

"Sum of all prime numbers is: {prime numbers sum}"

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

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 3  9  0  7  19  4  stop | Sum of all prime numbers is: 29  Sum of all non prime numbers is: 13 | Първото въведено число е 3. То е просто и го прибавяме съм сумата на простите числа.  Следващото число е 9. То не е просто и го прибавяме към сумата на непростите числа.  Числото 0 не е просто число и го прибавяме към сумата на непростите числа. Сумата става 9+0=9.  Следващите две числа са 7 и 19. Те са прости и всяко едно от тях го прибавяме към сумата на простите числа. 3+7=10 и 10+19=29.  Следва числото 4, което не е просто и го прибавяме към съответната сума 9+4=13.  Получаваме команда stop. Програмата прекъсва своето изпълнение и отпечатваме двете суми. | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 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 |

## Train the Trainers

Курсът **"**Train the trainers**"** е към края си и финалното оценяване наближава. Вашата задача е да помогнете на журито, което ще оценява презентациите, като напишете програма, в която да изчислява **средната оценка** от представянето на **всяка една презентация** от даден студент, а накрая **средният успех от всички тях**.

От конзолата на първият ред се прочита **броят на хората в журито** **n** - цяло число в интервала [1…20]

След това на отделен ред се прочита **името на презентацията** - текст

За всяка една презентация на нов ред се четат **n - на брой оценки** - реално число в интервала [2.00…6.00]

**След изчисляване на средната оценка за конкретна презентация,** на конзолата се отпечатва

**"{името на презентацията} - {средна оценка}."**

След получаване на **команда** **"Finish"** на конзолата се отпечатва **"Student's final assessment is {среден успех от всички презентации}."** и програмата **приключва**.

Всички оценки трябва да бъдат форматирани **до** **втория знак** **след десетичната запетая.**

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 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 – броят на хората в журито следователно ще получаваме по 2 оценки на презентация.  (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 | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 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. |

## Генератор за пароли

Да се напише програма, която чете две цели числа *n* и *l*, въведени от потребителя, и генерира **по азбучен ред** всички възможни **пароли**, които се състоят от следните **5 символа**:

* Символ 1: цифра от 1 до *n*.
* Символ 2: цифра от 1 до *n*.
* Символ 3: малка буква измежду първите *l* букви на латинската азбука.
* Символ 4: малка буква измежду първите *l* букви на латинската азбука.
* Символ 5: цифра от **1** до n, по-голяма от първите 2 цифри.

### Вход

Входът се чете от конзолата и се състои от **две** **цели числа** *n* и *l* в интервала [1…9], по едно на ред.

### Изход

На конзолата трябва да се отпечатат **всички пароли** по **азбучен ред**, разделени с **интервал**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2  4 | 11aa2 11ab2 11ac2 11ad2 11ba2 11bb2 11bc2 11bd2 11ca2 11cb2 11cc2 11cd2 11da2 11db2 11dc2 11dd2 |
| 3  1 | 11aa2 11aa3 12aa3 21aa3 22aa3 |
| 3  2 | 11aa2 11aa3 11ab2 11ab3 11ba2 11ba3 11bb2 11bb3 12aa3 12ab3 12ba3 12bb3 21aa3 21ab3 21ba3 21bb3 22aa3 22ab3 22ba3 22bb3 |
| 4  2 | 11aa2 11aa3 11aa4 11ab2 11ab3 11ab4 11ba2 11ba3 11ba4 11bb2 11bb3 11bb4 12aa3 12aa4 12ab3 12ab4 12ba3 12ba4 12bb3 12bb4 13aa4 13ab4 13ba4 13bb4 21aa3 21aa4 21ab3 21ab4 21ba3 21ba4 21bb3 21bb4 22aa3 22aa4 22ab3 22ab4 22ba3 22ba4 22bb3 22bb4 23aa4 23ab4 23ba4 23bb4 31aa4 31ab4 31ba4 31bb4 32aa4 32ab4 32ba4 32bb4 33aa4 33ab4 33ba4 33bb4 |

## Специални числа

Да се напише програма, която чете **едно цяло число** N, въведено от потребителя, и генерира всички възможни **“специални”** **числа** от **1111** до **9999**. За да бъде **“специално”** едно число, то трябва да отговаря на **следното условие**:

* **N да се дели на всяка една от неговите цифри без остатък.**

**Пример:** при **N = 16**, **2418** е специално число:

* **16 / 2** = 8 **без остатък**
* **16 / 4** = 4 **без остатък**
* **16 / 1** = 16 **без остатък**
* **16 / 8** = 2 **без остатък**

### Вход

Входът се чете от конзолата и се състои от **едно** **цяло число** в интервала **[1…600000]**

### Изход

На конзолата трябва да се отпечатат **всички “специални” числа**, разделени с **интервал**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 3 | 1111 1113 1131 1133 1311 1313 1331 1333 3111 3113 3131 3133 3311 3313 3331 3333 | 3 / 1 = 3 без остатък  3 / 3 = 1 без остатък  3 / 3 = 1 без остатък  3 / 3 = 1 без остатък |
| 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 | |