# СофтУниада 2020

## 01. Мечтана Кола

Автор: Тонислав Троев

Митко получава заплата в размер ***N*** лева. Всеки месец неговите разходи са точно ***M*** лева, но и заплатата му се увеличава с ***X*** лева. Мечтата на Митко е да се купи хубава кола, която обаче е прекалено скъпа за него (струва точно ***Y*** лева). Помогнете на Митко да изчисли дали след ***T*** месеца ще е спестил достатъчно пари, за да си позволи мечтания автомобил.

### Вход

* На първият ред получаваме числото **N** – стартовата заплатата на Митко.
* На вторият ред получаваме числото **M –** месечните разходи на Митко.
* На третия ред получаваме числото **X** – сумата, с която всеки месец заплатата нараства.
* На четвъртия ред получаваме числото **Y** – цената на мечтания автомобил.
* На петия ред получаваме числото **T** – броят месеци, през които Митко е спестявал пари.

### Изход

* На първият ред от изхода трябва да се отпечата **"Have a nice ride!"**, ако Митко е спестил достатъчно пари през тези **T** месеца. Иначе отпечатайте **"Work harder!"**.

### Ограничения

* Числото **N** е реално число в интервала **[10-10…100 000 000]**;
* Числото **M** е реално число в интервала **[10-10…100 000 000]**;
* Числото **X** е реално число в интервала **[10-10…100 000 000]**;
* Числото **Y** е реално число в интервала **[10-10…100 000 000]**;
* Числото **T** е цяло число в интервала **[1…12 000 000 000]**;
* Позволено време: **0.100с**
* Позволена памет: **16.00мб**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 100  50  10  500  7 | Have a nice ride! | За седем месеца Митко ще е изкарал:  100 + 110 + 120 + 130 + 140 + 150 + 160 = **910** лева  За същия период той ще е похарчил:  7 \* 50 = **350** лева  В края на краищата Митко е успял да спести:  910 – 350 = **560** лева (които са достатъчни, за да си купи колата на стойност от **500** лева) |
| 560.8  600.4  15.3  4356.19  24 | Work harder! | След две години Митко ще е спестил само **3272.4** лева, които няма да са му достатъчни, за да закупи мечтания автомобил. |
| 1500  500  13  25003  22 | Have a nice ride! |  |

## 02. Нова Сграда

Автор: Вероника Начева

На СофтУни им предстои да се преместят в нова сграда, която да побере всички жадни за знания студенти. Вътрешните дизайнери обаче са изправени пред проблем, а именно оптимизиране на местата в новите зали.

Тъй като те не могат да се справят сами и са наясно, че студентите на СофтУни могат да им помогнат без затруднение, дизайнерите ви възлагат задачата.

По зададени размери (цяло число) трябва да изчислим и нарисуваме на конзолата колко места могат да се поберат в съответната зала, имайки предвид следното:

* Седящите места трябва да са разположени по **диагонал** и означени с **'#'.**
* Разстоянието между местата (празно място) се означава с **'...'**.
* Чертежът на залата трябва да започне със седящо място в **горният ляв ъгъл** и да следва шаблона **стол-празно място-стол-празно място**.
* Седящите места (столове) са разположени **горе-дясно към долу-ляво**.
* За повече яснота вижте примерите по-долу.

### Вход

* **Входът се прочита от конзолата.**
* **Един ред – размера на залата – цяло число [3…151].**

### Изход

* **Изходът трябва да е чертеж на разположението на седящите и празни места в залата.**

### Ограничения

* Позволено време: **0.100с**
* Позволена памет: **16.00мб**

### Примери

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| 7 | #...#..  ...#...  ..#...#  .#...#.  #...#..  ...#...  ..#...#  .#...#.  #...#..  ...#... |  | 9 | #...#...#  ...#...#.  ..#...#..  .#...#...  #...#...#  ...#...#.  ..#...#..  .#...#...  #...#...#  ...#...#.  ..#...#..  .#...#...  #...#...# |  | 5 | #...#  ...#.  ..#..  .#...  #...#  ...#.  ..#.. |

## 03. Колода Карти

Автор: Тонислав Троев

Ники обича да си играе с колода от **N** на брой карти. За тази цел той всеки път ги разбърква хубаво, за да могат те да бъдат подредени по различен начин – той си избира произволно място **X** от тестето, където го разделя на две купчини, които смесва по следния начин – взема 1 карта от първата купчинка, след това 1 от втората и така, докато картите в една от двете купчини не свършат.

### Вход

* На първият ред получаваме числото **N** – броя на картите в колодата.
* На вторият ред получаваме няколко числа **X** –индексите от масива, в които тестето се разделя на две купчини.

### Изход

* На първият ред от изхода трябва да се отпечатат последователно всички карти от колодата на Ники в реда, в които те ще бъдат поставени след всички размесвания.

### Ограничения

* Числото **N** ще е цяло число в интервала **[1…1 000]**;
* Числата **X** ще са цели числа в интервала **[0…N - 1]**;
* Позволено време: **0.100с**
* Позволена памет: **16.00мб**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 5  3 3 2 | 1 4 5 3 2 | В началото имаме следната подредба: **[1, 2, 3, 4, 5]**  След първото размесване получаваме: **[1, 4, 2, 5, 3]**  След второто размесване получаваме: **[1, 5, 4, 3, 2]**  След третото размесване получаваме: **[1, 4, 5, 3, 2]** |
| 7  0 1 2 3 | 1 4 3 5 2 6 7 | В началото имаме следната подредба: **[1, 2, 3, 4, 5, 6, 7]**  След първото размесване получаваме: **[1, 2, 3, 4, 5, 6, 7]**  След второто размесване получаваме: **[1, 2, 3, 4, 5, 6, 7]**  След третото размесване получаваме: **[1, 3, 2, 4, 5, 6, 7]**  След четвъртото размесване получаваме: **[1, 4, 3, 5, 2, 6, 7]** |

## 04. Скачай

Автор: Стоян Шопов

Играл ли си някога играта "**Soft Jump**". Най-вероятно не, защото тя все още не съществува и познай какво… Твоята задача е да я създадеш.

На първия ред ще получиш две числа "**n**" и "**m**", които представляват редовете и колоните на полето. Полето съдържа три различни символа – "**0**", "**-**" и "**S**". "**S**" представлява **играча**, а той винаги стартира от **последния ред**. "**-**" е място, където играч може да **стъпи** върху него, а "**0**" е **празно** място.

Правилата на играта са прости. Играчът стартира от дъното на полето и започва да скача **нагоре**, докато не излезе от него. Разбира се, има уловка. Всички клетки, които съдържат "-" могат да се движат надясно **x** на брой пъти (ако стъпките са **повече** от **дължината**, просто се **превъртат** стъпките) .

Играча, **може** да скача **нагоре**, само ако целевата клетка съдържа "**-**". Ако се случи така, че играча се опита да скочи върху клетка, която съдържа "**0**", той трябва да остане на **същата** позиция.

Играчът скача само **след** получена команда за местене на редовете.

Играта **приключва**, когато играчът скочи през **всички** редове (входните данни, валидират, че играча винаги излиза от полето).

### Вход

* На първия ред, входа ще се състои от две числа – **n** (редове) и **m** (колони)
* На следващите **n** реда, ще получавате стрингове с дължина **m**, които ще съдържат – "**0**", "**-**" или "**S**". Примери – "**-000**", "**00S0**"
* Ще получите **k** брой на редовете, които трябва да се прочетат
* На следващите **k** редове ще получаваш **координати**, които представляват определен **ред** и брой **стъпки**, с които трябва да се завъртят **надясно** колоните.

### Изход

* На първия ред: "**Win**" или "**Lose**"
* На втория ред: "**Total Jumps:** {**брой подскоци**}"
* На следващите редове: игралното поле във финалният си вид

### Ограничения

* Редовете и колоните ще бъдат винаги между **3** и **10**
* Стъпките ще бъдат между **0** и **10000000**
* Всички получени данни от конзолата ще бъдат **валидни**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснение** |
| 5 4  00-0  0-00  0-00  -000  00S0  4  3 2  2 1  1 1  0 0 | Win  Total Jumps: 4  00S0  00-0  00-0  00-0  0000 | Трябва да създадете поле, което с размери **5** реда и **4** колони. Играчът стартира на координати **4**,**2**. **След** всяка команда за местене на колоните, играча се опитва да **скочи** на следващия ред. **3** **2** означава, че на **3-ти** **ред** трябва да преместим клетките с **2 стъпки** надясно. |
| 5 3  00-  -00  0-0  00-  0S0  4  3 2  2 0  1 1  0 2 | Win  Total Jumps: 4  0S0  0-0  0-0  0-0  000 |  |
| 4 4  00-0  0-00  00-0  S000  3  2 1  2 1  1 1 | Lose  Total Jumps: 1  00-0  00-0  S000  0000 |  |
| 5 5  -000-  00-00  0-0-0  -000-  S0000  2  3 1  2 2 | Lose  Total Jumps: 2  -000-  00-00  S00-0  --000  00000 |  |
| 6 6  0--000  00--00  000--0  0000--  0000--  000S00  10  4 1  3 2  2 3  1 4  0 5  4 3  3 2  2 2  1 2  0 2 | Win  Total Jumps: 5  00-S00  00--00  00--00  00--00  00--00  000000 |  |

## 05. Път

Автор: Ивайло Кенов

Прибираш се по тъмно и изведнъж се оказваш изгубил се без батерия на телефона. За щастие имаш някаква "**псевдо**" карта, с която можеш да се ориентираш. "**Псевдо**", защото всъщност картата ти е просто посоки на завиване. Възможните посоки са **напред** ("S"), **наляво** ("L") и **надясно** ("R"). Съответно – картата изглежда нещо подобно: **LSRLRSRLLR**, което означава – завий наляво, продължи направо, завий надясно, завий наляво, завий надясно и т.н. докато не си вкъщи. Или поне така изглеждаше картата, преди майка ти да ти изпере панталоните заедно с нея. В момента картата изглежда покъртително, но все пак можеш да прочетеш част от символите. Останалите символи ги маркираме с "**\***". Тоест картата сега може да изглежда така: **LR\*\*SR\*LL**, което означава – завий наляво, завий надясно, всички посоки са възможни, всички посоки са възможни пак, продължи направо, завий надясно, всички посоки са възможни и т.н. докато не си вкъщи. На практика всяко "**\***" може да бъде "**S**", "**L**" или "**R**". Задачата ти е да намериш всички възможни пътища, които биха могли да се формират от нечетимата карта.

### Вход

* На първият ред получаваме частичната карта – текст, съдържащ символите "**S**", "**L**", "**R**" и "**\***".

### Изход

* На първият ред от изхода трябва да се отпечата **N** броят на уникалните възможни пътища.
* На следващите **N** реда трябва да се отпечатат всички уникални възможни пътища, сортирани по азбучен ред.

### Ограничения

* Дължината на пътя ще бъде максимум 16 символа.
* Позволено време: **0.100с**
* Позволена памет: **16.00мб**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| LSLLRSRL | 1  LSLLRSRL |
| R\*S\*L | 9  RLSLL  RLSRL  RLSSL  RRSLL  RRSRL  RRSSL  RSSLL  RSSRL  RSSSL |
| \*\*RLR\* | 27  LLRLRL  LLRLRR  LLRLRS  LRRLRL  LRRLRR  LRRLRS  LSRLRL  LSRLRR  LSRLRS  RLRLRL  RLRLRR  RLRLRS  RRRLRL  RRRLRR  RRRLRS  RSRLRL  RSRLRR  RSRLRS  SLRLRL  SLRLRR  SLRLRS  SRRLRL  SRRLRR  SRRLRS  SSRLRL  SSRLRR  SSRLRS |

## 06. Минимална Неравност

Автор: Вероника Начева

Дадени са 2 цели числа: ***k*** и ***n***, които трябва да се прочетат на **2 отделни реда**. На следващите редове ще бъдат подадени елементите на списък с дължина ***n***.

Трябва да се създаде втори лист с **дължина *k***, такъв че елементите му да бъдат с **минимална неравност**. Минимална неравност означава, че разликата между **най-голямото** и **най-малкото** число в **подсписъка** с дължина ***k*** трябва да бъде **минимална** (най-малката възможна).

Например, в списъкът **[1, 4, 7, 2],** с ***k* *= 2*,** подсписъкът с минимална неравност би бил **[1, 2]** (тъй като разликата между тях е минималната възможна от всички други двойки).

***Бележка: Възможно е даденият списък да съдържа повтарящи се елементи.***

### Вход

* На първият ред ще получите числото **K** **[2, N]**
* На вторият ред ще получите числото **N** **[2, 10⁵]**
* На следващите **N реда** ще получите самите **числа**

### Изход

* **Цяло число**, което представлява **минималната разлика** между **най-големият** и **най-малкият** елемент на **подсписъка**

### Ограничения

* Позволено време: **0.300с**
* Позволена памет: **19.00мб**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 3  7  10  100  300  200  1000  20  30 | 20 | Тук трябва да създадем подсписък с дължина 3. В този случай [10, 20, 30] с разлика между най-големия елемент и най-малкия 20 е минималната възможна за дадените числа. |
| 4  10  1  2  3  4  10  20  30  40  100  200 | 3 | Тук трябва да създадем подсписък с дължина 4. В този случай [1, 2, 3, 4] с разлика 3 е минималната възможна с дадените числа. |
| 2  5  1  2  1  2  1 | 0 |  |

## 07. Камиони

Автор: Виктор Даков

Ники решава да си отвори фабрика за наргилета в родния си град. Той има само един клиент, който се намира в друг град и иска да купи колкото може повече. За целта Ники трябва да натовари камиони пълни с наргилета и да ги изпрати по пътищата.

Ще получите името на града, от който всички камиони тръгват. Ще имате и града, до който всички превозни средства трябва да стигнат. Ще получите n на брой **пътища** между два **града** и **количество** на камиони, които могат да се движат по пътя. Всеки път си има определена **посока** на движение. Помогнете на Ники да намери максималният брой камиони, които могат да бъдат изпратени от началния град, до града на клиента. Важно е в нито един момент да няма повече камиони на пътя от възможното за този път **количество.**

### Вход

* На първият ред получаваме името на началния ни град
* На втория ред получаваме името на крайният (целевия) град
* На третия ред получаваме едно число N
* На следващите N реда получаваме три x y и z. Посоката на движение е x -> y
  + x e името на града от който започва пътят
  + y е името на града, където пътят свършва
  + z е число, обозначаващо максималният брой камиони, които пътят събира

### Изход

* На първият ред от изхода трябва да се отпечата **N** броят на уникалните възможни пътища.
* На следващите **N** реда трябва да се отпечатат всички уникални възможни пътища, сортирани по азбучен ред.

### Ограничения

* **N** <= **1000**
* **Z** <= **1000000**
* Позволено време: **0.100с**
* Позволена памет: **16.00мб**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| sofia  varna  9  sofia pazardjik 10  sofia velingrad 10  pazardjik velingrad 2  pazardjik plovdiv 8  velingrad plovdiv 9  pazardjik lovech 4  lovech varna 10  plovdiv lovech 6  plovdiv varna 10 | 19  // От град софия могат да тръгнат най-много 19 коли  // така че на никой път да няма повече от  // допустимото превзони средства |

## 08. Пръчки

Автор: Ивайло Кенов

Дадени са ни **N** на брой пръчки с различни размери за дължина и широчина (размерите могат да се **повтарят**).

Напишете програма, която намира **всички** **уникални** начини, по които можем да наредим тези пръчки като им сменяме размерите и ги обръщаме.

Например – ако имаме 3 пръчки с размери |2-3|, |2-2|, |3-2|, можем да ги подредим по следните 12 начина:

|2-2| # |2-3| # |2-3|

|2-2| # |2-3| # |3-2|

|2-2| # |3-2| # |2-3|

|2-2| # |3-2| # |3-2|

|2-3| # |2-2| # |2-3|

|2-3| # |2-2| # |3-2|

|2-3| # |2-3| # |2-2|

|2-3| # |3-2| # |2-2|

|3-2| # |2-2| # |2-3|

|3-2| # |2-2| # |3-2|

|3-2| # |2-3| # |2-2|

|3-2| # |3-2| # |2-2|

### Вход

* На първият ред получаваме числото **N** – броят на пръчките.
* На следващите **N** реда, получаваме размерите на всяка една от пръчките.

### Изход

* На първият ред от изхода трябва да се отпечата **K** броят на уникалните подреждания на пръчките.
* На следващите **K** реда трябва да се отпечатат всички уникални подредби на пръчките, сортирани лексикографски във форматът посочен в примера.

### Ограничения

* Всички числа във входа ще са цяло число в интервала **[1…6]**;
* Позволено време: **0.900с**
* Позволена памет: **27.00мб**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  2 3  2 2  3 2 | 12  |2-2| # |2-3| # |2-3|  |2-2| # |2-3| # |3-2|  |2-2| # |3-2| # |2-3|  |2-2| # |3-2| # |3-2|  |2-3| # |2-2| # |2-3|  |2-3| # |2-2| # |3-2|  |2-3| # |2-3| # |2-2|  |2-3| # |3-2| # |2-2|  |3-2| # |2-2| # |2-3|  |3-2| # |2-2| # |3-2|  |3-2| # |2-3| # |2-2|  |3-2| # |3-2| # |2-2| |

## 09. Нови Монети

Автор: Николай Костов

СофтУни въвежда нова парична система, която използва монети със следните стойности: 1, 10, 25, 100, 1000, 2500, 10000, 100000, 250000, 1000000, 10000000, 25000000 и така нататък. С други думи за всяко **K >= 0** има монети, които са със стойност **10K**и монети със стойност **25\*100K**.

Новата яка тениска на СофтУни струва **P**. Да приемем, че ти си достатъчно богат и имаш неограничен брой монети от всеки тип. Намери най-малкия брой монети, с които можеш да платиш точната цена на тениската (**P**).

### Вход

На единствения ред на входа ще бъде числото **P**.

### Изход

Принтирай най-малкия брой от монети, които са нужни, за да се плати точно цената **P.**

### Ограничения

* **P** ще бъде цяло число в интервала [1; 1,000,000,000,000,000].
* Позволено време: **0.100с**
* Позволена памет: **16.00мб**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснение** |
| 39 | 6 | 25+10+1+1+1+1 |
| 8 | 8 | 1+1+1+1+1+1+1+1 |
| 30 | 3 | 10+10+10 – 3 монети  Има и друго решение тук, но то има 6 монети: 25+1+1+1+1+1 |
| 2772788690199 | 36 |  |

## 10. Потребителски Имена

Автор: Николай Костов

СофтУни са решили да сменят изискванията за потребителското име в сайта им softuni.bg.

Новите потребителски имена ще са с дължина **точно N символа**. Трябва да съдържат **поне D цифри** (има 10 възможни цифри 0-9), **поне L малки български букви** (има 30 различни български малки букви [а-я]), и **поне U главни български букви** (има 30 различни български главни букви [А-Я]).

Помогни на СофтУни да намери броя на всички възможни потребителски имена, отговарящи на съответните правила. Изведете отговора на конзолата, модулно разделен на 1,000,000,007.

### Вход

На първия ред ще получите числото **N**. На втория ред ще получите числото **D**.

На третия ред ще получите числото **L**. На четвъртия ред ще получите числото **U**.

### Изход

На единствения изходен ред напишете броя на всички възможни потребителски имена по модул 1,000,000,007.

### Ограничения

* **N** ще бъде цяло число в интервала [1; 100,000].
* **D**, **L** и **U** ще бъдат цели числа в интервала [0; **N**].
* Позволено време: **0.500с**
* Позволена памет: **18.00мб**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 2  2  0  0 | 100 | Тук потребителските имена трябва да бъдат с дължина 2 и да съдържат поне 2 цифри, така че единствените валидни потребителски имена са "00" – "99" и има 100 възможни потребителски имена. |
| 3  1  1  1 | 54000 | Отговорът е 3! \* 10 \* 30 \* 30. За щастие, в този случай точният брой на символите от конкретен тип е известен. Валидно потребителско име съдържа точно една цифра, точно една малка буква и точно една главна буква. |
| 10  4  4  4 | 0 | Този случай не удовлетворява изискванията. |
| 10  3  1  3 | 691232721 | Тук символите са едва 10, а отговорът доста по-голям. |