НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
Департамент Информатика
Отборно състезание по програмиране, 20.04.2008

Задача А. Минимално и максимално n-цифрено число

Дадено е едно m-цифрено цяло положително число N. За всяко цяло число n ($0 < n \leq m$) да се намерят най-малкото и най-голямото n-цифрени числа, които могат да се получат от цифрите на числото N, без да се разменят местата на никои две цифри, т.е. числата се получават само чрез изтриване на някои цифри на числото N.

Стандартен вход

Всеки пример се задава с едно m-цифрено число ($2 \leq m \leq 20$). Входът съдържа няколко примери и на последният ред съдържа числото 0 за край на вход.

Стандартен изход

За всеки пример от входа на изхода се записват m реда. Всеки ред съдържа две числа – най-малкото и най-голямото намерени n-цифрени числа за $n = 1, 2, \ldots, m$. След всеки пример се оставя по един празен ред.

Пример:

Вход

```
12
541623
0
```

Изход

```
2 1
12 12

6 1
63 12
623 123
5623 1623
54623 41623
541623 541623
```

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

Департамент Информатика

Отборно състезание по програмиране, 20.04.2008

Задача B. Единици и нули

Даден низ, съставен само от цифрите 1 и 0, съдържащ не повече от 10^6 цифри. Да се отговори на въпроса дали поднизът, започващ от символ с индекс i и завършващ със символ с индекс j ($i \leq j$) се състои само от единици или само от нули.

Стандартен вход

Първият ред от всеки пример задава низ от единици и нули. Следващият ред съдържа броя n на въпросите в този пример. Следващите n реда съдържат въпросите, дадени с по две неотрицателни числа, i и j. Входът съдържа няколко примери.

Стандартен изход

За първия пример от входния файл в изходния файл се записва `Case 1:` и на следващите редове – отговорите на въпросите `Yes` или `No`. За следващите примери се пише следващ номер на пример и на следващите редове пак “верните отговори”.

Пример:

Вход

```
0000011111
3
0 5
2 4
5 9
0101010101010101010101010111111111111111111111111111111111111110000000000000000
4
4 4
25 60
1 3
62 76
1
1
0 0
```

Изход

```
Case 1:
No
Yes
Yes
Case 2:
Yes
Yes
No
Yes
Case 3:
Yes
```

Задача C. Специални числа

Да се намери броят на 10-цифрените числа, в които всяка цифра участва точно по веднъж и сумата на първите 5 цифри е по-малка от сумата на последните 5 цифри. Програмата няма вход, а изходът е едно число.

Задача D. Simply Subsets

After graduating from the University of Notre Dame, you obtained a job at Top Shelf Software, Inc., as an entry-level computer engineer. On the first day, your manager sits down with you and tasks you with the following job: "We want to see how well you understand computer programming and the abstract science behind it. As an evaluation for all of our new hires, we require them to write a program to determine the relationship between pairs of sets. I'm quite sure that you'll do well; my confidence is high. Here's a list of requirements for what the program should do. Good luck."

Input

Your program should accept an even number of lines of text. Each pair of lines will represent two sets; the first line represents set `A`, the second line represents set `B`. Each line of text (set) will be a list of distinct integers.

Output

After each pair of lines has been read in, the sets should be compared and one of the following responses should be output:

```
A is a proper subset of B
B is a proper subset of A
A equals B
A and B are disjoint
I'm confused!
```

Sample Input

```
55 27
55 27
9 24 1995
9 24
1 2 3
1 2 3 4
1 2 3
4 5 6
1 2
2 3
```

Sample Output

```
A equals B
B is a proper subset of A
A is a proper subset of B
A and B are disjoint
I'm confused!
```

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
Департамент Информатика
Отборно състезание по програмиране, 20.04.2008

Задача E. Римски и арабски цифри

Римските цифри са цифри, използвани от древните римляни в тяхната непозиционна бройна система. Те са:

I – 1, V – 5, X – 10, L – 50, C – 100, D – 500, M – 1000

Изписването на числата с римските цифри се базира на няколко основни принципа:

* Всеки символ, намиращ се отдясно на друг символ с по-голяма стойност, се прибавя към тази стойност.

* Всеки символ, намиращ се отляво на друг символ с по-голяма стойност, се изважда от тази стойност.

* Символите са групирани в низходящ ред по стойност освен тези, за които се прилага предишното правило. На практика това правило гласи, че при римските цифри първо се изписват хилядните, после стотиците, след това десетиците и накрая единиците.

* Символ, представляващ стойност 10^n, не може да се поставя пред символ, по-голям от 10^{n+1}. Така например M може да бъде предшестван единствено от C, но не от I, V или X.

Арабските цифри са десет знака, които се използват за записването на числа в позиционна десетична бройна система. Арабските цифри са:

1, 2, 3, 4, 5, 6, 7, 8, 9 и 0.

Арабските цифри произхождат от индийски символи за записване на числата. В Европа добиват популярност през X-XIII в. Днес арабските цифри се използват в цял свят.

Да се преобразува запис на число с римски цифри в запис на същото число с арабски цифри. На входа се задават числа, написани с римски цифри, по едно число на ред. За всяко число от входа да се изведе числото, написано с арабски цифри в десетична бройна система, пак по едно на ред.

Пример:

Вход	Изход
`IX`	`9`
`XVI`	`16`
`MCMLXXXIX`	`1989`

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

Департамент Информатика

Отборно състезание по програмиране, 20.04.2008

Задача F. Произведение

Дадена е редица от естествени числа $a_1, a_2, \ldots, a_n$. Да се намери броят на нулите, на които завършва произведението $p = a_1 a_2 \ldots a_n$.

Вход

Входът се състои от много примери, като на всеки ред се задава по една редица от цели положителни числа (не повече от 100 и не по-големи от 1000).

Изход

За всеки ред от входа, на изхода се отпечатва броят на нулите, на които завършва произведението.

Пример:

Вход	Изход
`5 2 3`	`1`
`22 10 55`	`2`
`1 2 3 4 5 6 7 8 9 1000`	`4`

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
Департамент Информатика
Отборно състезание по програмиране, 20.04.2008

Задача G. Опашка

В повечето банкови институции, за по-доброто обслужване на клиентите, вече са инсталирани компютризирани системи. Една такава система регистрира пристигналия в банката клиент, подрежда го в опашка и, когато се появи възможност за обслужване, извиква от опашката клиента, който е наред да бъде обслужен. Затова и новосъздадената Първа Приоритетна Банка (ППБ) решила да внедри подобна система, като отчете някои свои особености. Клиентите на ППБ се идентифицират с постоянен клиентски номер k и при всяко свое идване в банката получават различен приоритет за обслужване p, с който се нареждат на опашката. Когато дойде време за обслужване, от опашката се извиква клиентът с най-голяма стойност на приоритета в момента. Като на всяка опашка, клиентите чакащи за обслужване нервничат. Не рядко някой от чакащите се обръща към обслужващия персонал и пита "Колко клиенти има преди мен на опашката в момента?", а ако в резултат получи твърде голямо число, може да реши и да се откаже да чака повече. Като програмист на банката трябва да реализирате програмно основна част от бъдещата обслужваща система.

Вход

Програмата трябва да обработва заявки от следния вид:

1	k p	– клиентът с номер k се нарежда на опашката с приоритет p;
2		– клиентът с най-голям приоритет се изважда от опашката;
3	k	– клиентът с номер k пита колко души има преди него в опашката;
4	k	– клиентът с номер k напуска опашката без да бъде обслужен,

като $1 \leq k \leq 10^6$, $1 \leq p \leq 10^7$. Заявките се четат от стандартния вход, като на всеки ред е записана по една заявка. Последният ред на входа съдържа само 0. Не е възможно в заявка от тип 1 да се появи номер или приоритет, които вече се намират в опашката, нито пък в заявки от тип 3 и 4 - номер на клиент който не е в опашката.

Изход

За всяка заявка от тип 2 програмата трябва да изведе на отделен ред на стандартния изход номера на клиента, който е изваден от опашката за обслужване, а ако опашката е празна – да изведе 0. За всяка заявка от тип 3 програмата трябва трябва да изведе на отделен ред на стандартния изход търсения брой клиенти.

Пример:

Вход	Изход
`1 5 1000`	`0`
`3 5`	`1`
`1 1 1002`	`1`
`3 5`	`6`
`1 6 300`	
`2`	
`4 5`	
`2`	
`0`	

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
Департамент Информатика
Отборно състезание по програмиране, 20.04.2008

Задача H. Фибонод

Всички знаете кой е Станчо (той се изучава още в училище). По-малко известни в научните среди, обаче, са имената Евклид и Леонардо Писано, наричан Фибоначи. (Не сме сигурни какво е малкото име на Евклид, но се предполага, че се е казвал Димитър.) Въпреки, че нямат приноса на Станчо, те са изиграли важна роля в развитието на математиката през миналото. Ето защо Станчо смята, че те трябва да бъдат уважавани и изучавани в училищата.

Тъй като Станчо е основна фигура в министерството на образованието, по негово настояване бе издадена наредба, според която всички ученици след трети клас трябва да държат матура по програмиране. За целта той е измислил задача изискваща познаването на постиженията на Фибоначи и Евклид. Вашата работа, е да напишете авторското решение на тази задача. Разбира се, разгласяването на условието на тази задача на външни лица или претенцияа за авторство на решението е забранено и би довело до неприятни стечения на обстоятелствата за вас. Числата на Фибоначи се дефинират по следния начин

$F(0) = 0, F(1) = 1, F(i) = F(i-1) + F(i-2)$ при $i > 1$.

Най-голям общ делител на две естествени числа a и b, означаваме го с НОД(a, b), наричаме такова цяло положително k, което дели a и b без остатък и няма друго, по-голямо от него, цяло със същото свойство. Предполага се, че знаете тези неща от кръстословиците (както и, че можете да ги пресмятате с експоненциална сложност). Напишете програма, която по зададени цели a и b $(0 < a, b \leq 2000)$ да пресмята НОД$(F(a), F(b))$.

Вход

Входните данни се четат от стандартния вход. На всеки ред от него има по една двойка числа a и b. Данните завършват с двойката 00, която не трябва да се обработва.

Изход

За всяка двойка от входните данни програмата трябва да изведе на отделен ред на стандартния изход НОД$(F(a), F(b))$.

Пример:

Вход	Изход
`4 6`	`1`
`6 3`	`2`
`0 0`	

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

Департамент Информатика

Отборно състезание по програмиране, 20.04.2008

Задача I. Прости числа

Има безкрайно много прости числа. Най-старото известно доказателство на този факт е дадено от гръцкия математик Евклид в книгата му Елементи. Твърдението на Евклид е:

Броят на простите числа е по-голям от всяко отнапред зададено [крайно] число
и неговото доказателство по същество е следното:

Да допуснем, че множеството на простите числа е крайно и има m на брой елемента. Да умножим всички m прости числа и към резултата да добавим едно. Тъй като полученото число е по-голямо от всяко просто, то не принадлежи на горното множество. Освен това, то не се дели на нито едно от крайния брой прости, защото ако го разделим с частно и остатък на някое от тях, ще получим остатък едно, а едно не се дели на никое просто число. Следователно то трябва или да е просто, или да се дели на някое просто число, което не принадлежи на горното множество. И в двата случая получаваме, че броят на всички прости числа трябва да бъде поне $m+1$, което е в противоречие с първоначалното допускане. Това означава, че допускането ни не е вярно, тоест има безбройно много прости числа.

Постулат на Бертран: Ако n е положително цяло число, по-голямо от 1, то винаги има просто число p, за което $n < p < 2n$. При зададено n да се намери броя на простите числа в интервала $(n, 2n)$.

Пример:

Вход	Изход
`3`	`1`
`20`	`4`
`100`	`21`