XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача А. Променливи

Дадени са n променливи a_1, a_2, ..., a_n. За всяко фиксирано k, $1 \le k \le n$, да образуваме израза, който е сума от всички различни произведения на k променливи, взети измежду дадените. Например при $n = 3$ и $k = 2$ този израз е $a_1a_2+a_1a_3+a_2a_3$. Напишете програма, която да намери най-голямата по модул m стойност на всички такива изрази за $k = 1,2,...,n$, при зададени стойности на променливите.

Вход

Програмата трябва да прочете от стандартния вход няколко (не повече от 10) тестови примера. Всеки тестов пример съдържа стойностите n ($0 < n < 12345$), m ($1 < m < 12345678$) и a_1, a_2, ..., a_n (a_i – цели, $0 < a_i < 12345$, $i = 1, 2, \ldots, n$). Програмата трябва да спре, когато прочете на мястото на n и m стойности, равни на нула.

Изход

За всеки тестов пример програмата трябва да изведе на отделен ред на стандартния изход най-голямата по модул m стойност на сумите от разглеждания вид.

Пример

Вход:	**Изход**:
3 1000000 1 2 3 4 10 1 2 3 4 0 0	11 5

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача В. Ребус

Една кръстословица представлява правоъгълна мрежа от квадратчета, някои от които са бели, а другите – черни. В белите се вписват думите на кръстословицата, а черните служат за разделители между думите. Думите се вписват "водоравно" от ляво надясно или "отвесно" от горе надолу като започват от специално номерирани за целта квадратчета. Номерират се всички бели квадратчета, които отговарят на някое от следните условия:

А) принадлежат на първия, най-горен ред;
Б) принадлежат на първия, най-ляв стълб;
В) квадратчето, с което имат обща страна и се намира непосредствено отляво или отгоре, е черно.

Номерацията се извършва така, че когато мрежата се обхожда последователно отляво надясно и отгоре надолу, с начало в горния ляв ъгъл, номерата да бъдат последователни числа, започващи от едно. Напишете програма, която:

- номерира необходимите бели квадратчета по описания начин;
- попълва кръстословицата "водоравно" със зададени думи;
- по зададен номер на квадратче извежда думата, която започва от това квадратче и е разположена "отвесно".

При попълване на "водоравните" думи може да се получи квадратче, в което няма записана буква. Ако такова квадратче се срещне при извеждане на желаната "отвесна" дума, то в съответната позиция се извежда знакът *.

Вход

От стандартния вход се въвеждат краен брой тестови примери. Входните данни за всеки тестов пример са разположени по следния начин. На първия ред, разделени с интервал, са зададени две цели числа n и m, ($2 \le n, m \le 20$), задаващи съответно брой редове и брой стълбове на кръстословицата. Следват редове с по две цели числа, разделени с интервал, задаващи координатите – номер на ред и стълб – на черните квадратчета. Въвеждането на координатите на черни квадратчета завършва с ред, съдържащ две числа минус едно, съответно за номер на ред и стълб. Следват редове, започващи с номера на квадратче, последван от думата, която тябва да се попълни "водоравно" от този номер. За индикация на край на въвеждането на думи служи ред, съдържащ нула. Последният ред на тестовия пример съдържа номера на квадратче, от което да започва търсената "вертикална" дума.

Изход

За всеки тестов пример програмата трябва да изведе на отделен ред на стандартния изход намерената "вертикална" дума.

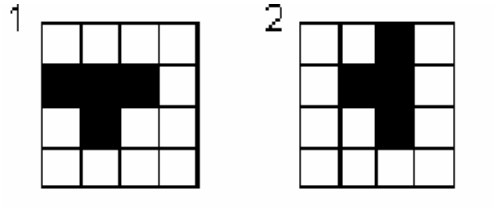
Пример

Вход	Изход
6 7 1 4 2 1 2 4 3 5 4 3 5 1 5 4 5 7 6 4 -1 -1 1sol 4rak 7do 8ana 9tema 11ar 12os 13rana 15ol 17la 18asa 19osa 0 2 6 7 1 4 2 1 2 4 3 5 4 3 5 1 5 4 5 7 6 4 -1 -1 1sol 4rak 7do 8ana 9tema 11ar 12os 13rana 15ol 17la 18asa 19osa 0 14	odesos alo

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача С. Ключ и ключалка

Известни са формата на ключ и ключалка, зададени като квадратни двумерни таблици от 0 и 1 (отворът на ключалката и сечението на ключа се задават с 1). Да се напише програма, която проверява дали зададеният ключ съответства на ключалката, като ключът може да е завъртян съответно на 0º, 90º, 180º и 270º (ключът може да се върти, но не може да се мести по хоризонтал и вертикал). Например, ключът от Рис. 1 може да съвпадне с ключалката от Рис. 2, ако бъде завъртян на 90º.



Вход

От стандартния вход се въвежда цяло положително число k – брой тестове. За всеки тест се въвежда цяло положително число n ($n \leq 10$) – размер на таблиците за ключа и ключалката. Следват n реда с по n числа (0 или 1) за ключа и n реда от по n числа (0 или 1) за ключалката.

Изход

За всеки тестов пример програмата трябва да изведе на отделен ред на стандартния изход резултата – `YES`, ако ключът съвпада с ключалката и `NO`, ако ключът не съвпада с ключалката.

Пример

Вход	**Изход**
2 4 0000 1110 0100 0000 0010 0110 0010 0000 3 000 010 011 000 100 110	YES NO

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача D. Кредитни карти

Петър и Павел написали програма за генериране на номера на кредитни карти, но не били сигурни дали работи вярно. Напишете програма, която проверява дали резултатът от програмата на Петър и Павел е валиден номер на кредитна карта VISA или MasterCard.

Номерата на кредитните карти са 16-цифрени, разделени на групи от по 4 цифри, като всяка група е разделена от съседните ѝ с тире. Проверката за валидност на картата се извършва по следния алгоритъм:

1. Цифрите на номера се вземат в обратен ред и се премахват тиретата;
2. Цифрите, стоящи на четни места се удвояват. Ако след удвояване на цифра се получи 2-цифрено число, то цифрата се замества с двете цифри на числото;
3. Намира се сбора на всички получени цифри;
4. Ако полученият сбор се дели на 10 без остатък, то номерът на картата е валиден. Ако номерът започва с 4, то типът ѝ е VISA, а ако има префикс от 51 до 55 е MasterCard.

Вход

От стандартния вход се прочита цяло число k – броя номера на кредитни карти, които трябва да бъдат проверени. От всеки от следващите редове се прочита по един номер на карта.

Изход

За всеки един от номерата програмата трябва да изведе на отделен ред на стандартния изход `VISA`, ако това е валиден номер на VISA карта, `MasterCard` – ако е валиден номер на Master Card, `YES` – ако е валиден номер, който не е нито VISA, нито Master Card и `NO` – ако не е валиден номер на карта.

Пример

Вход	**Изход**
3 4204-5876-9012-5234 5173-1249-4804-2492 8699-3855-7635-7380	VISA MasterCard NO

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача Е. Охрана

Охранителна фирма трябва да оранява n обекта ($n < 15$), номерирани с числата от 1 до n. За всеки ден се изготвя график, в който е записано кой от служителите на фирмата кой номер обект охранява. Графикът е низ, в който всеки служител е отбелязан с по една латинска буква – първата буква от името му, като охранителят назначен на първия обект е на първо място, охранителят назначен на втория обект е на второ място и т.н. Някои от охранителите са означени с една и съща буква. След оповестяване на графика тези от служителите, които са отбелязани с една и съща буква, се договарят кой от тях, кой обект да отиде да охранява. Графикът за всеки следващ ден се получава от графика на предишния, така че е следващият лексикограски спрямо предишния, получен от същите букви. Ако не е възможно да се направи следващ лексикографски, тогава се преминава към лекскографски най-малкия график. На служител със зададен инициал възниква проблем за вземане на дежурство след d дни. Помогнете на потенциалния му заместник, като отпечатате графика за съответния ден и намерите в него номерата на обекти, които заместникът би могъл да охранява.

Вход
От първия ред на стандартния вход се въвежда k ($k \le 31$) – броя на тестовете. В първия ред за всеки тест е зададен графикът с който се започва. На втория ред е зададен инициал x на служител, на който се налага да отсъства, а на третия – числото d. Инициалите се изписват само с малки латински букви. Търсеният инициал винаги съществува.

Изход
За всеки тест се извежда на един ред графика за след d дни, а на следващ ред – номерата на обекти, които ще се охраняват от служител с инициала x. Следващият лексикографски низ за график, състоящ се от еднакви буквии, е същият низ.

Пример

Вход	**Изход**
1 acbc c 1	accb 2 3

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача F. Покритие

Дадени са n отсечки върху една права ($0 < n < 100$). Всяка отсечка е зададена с координатите на двата си края a_i и b_i, $i = 1, ..., n$. Координатите са цели числа от интервала (−999, 999). Някои от отсечките могат да се припокриват. Напишете програма, която премахва минимален брой от дадените отсечки така, че никои две от останалите отсечки да нямат обща вътрешна точка, т.е. точка, която принадлежи едновременно на двете отсечки и е вътрешна за поне едната от тях.

Вход
Входните данни се четат от стандартния вход. От първия ред на стандартния вход се въвежда цяло число k – броя на тестовите примери. За всеки тестов пример данните се въвеждат по следния начин: От първия ред се въвежда цялото число n. Следват n реда, всеки съдържащ по две цели числа, разделени с един интервал. Всяка от тези двойки числа задава координатите на двата края на поредната от дадените отсечки.

Изход
Резултатът от програмата трябва да бъде записан на стандартния изход, като за всеки тестов пример се извежда единствено цяло число, равно на броя на отсечките, които са останали, след като програмата е премахнала тези, за които това се изисква от условието на задачата.

Пример

Вход	**Изход**
2 3 6 3 1 3 2 5 6 1 30 1 2 2 3 4 5 5 6 100 200	2 5

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

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 $1 \le k \le 10^6$, $1 \le p \le 10^7$. 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 3 5 1 1 1002 3 5 1 6 300 2 4 5 2 0	0 1 1 6

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача Н. Еднопосочно движение

Улиците и кръстовищата на града *X* са такива, че от всяко кръстовище може да се стигне до всяко друго кръстовище. Кръстовищата са номерирани с числата от 1 до *n*, а всяка от *m*-те улици се определя с двата номера на кръстовища, които свързва директно. Градът е разположен на основния път от София за Варна така, че пътят влиза в града през кръстовището с номер 1 и излиза през кръстовището с номер *n*. Всичко в града беше наред до неотдавна, когато кметът реши, за да облекчи движението на превозни средства в града, да направи всички улици еднопосочни (ах тези кметове!!!). Оказа се, че при такава реформа движението между кръстовищата 1 и *n* може да се окаже невъзможно (или заради липсата на път от 1 до *n*, или заради липсата на път от *n* до 1, или и двете). Тежестта пак легна върху плещите на програмиста на кметството. Той трябва да напише програма, която, при избраната от кмета ориентация на улиците да намери минимален брой улици, които да останат двупосочни, така че пътуващите между София и Варна да могат безпрепятствено да напуснат града.

Вход
На първия ред на стандартния вход ще бъде зададен броят *k* на тестовите примери, които програмата трябва да реши. Всеки тест започва с ред, на който са дадени целите числа *n* и *m* ($3 \le n \le 1000$). На всеки от следващите *M* реда са дадени по два номера на кръстовище, свързани директно с улица, като според идеята на кмета посоката на движението трябва да е от първото към второто кръстовище.

Изход
За всеки тестов пример програмата трябва да изведе на стандартния изход ред с намерения минимален брой улици, които трябва да останат двупосочни.

Пример

Вход	**Изход**
2 4 3 1 2 2 3 3 4 4 4 1 2 1 3 4 2 4 3	3 2

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача I. Двойна серия

Много от вас са чували за Станчо от Клуба на любителите на минерална вода. Освен с рекордните количества минерална вода, които може да изпие, обаче, в университетските сред той е известен и с разнообразни научни постижения. Тъй като Станчо има постижения в почти всички области на науката, алгоритмите с низове не правят изключение. Поради тази причина той е решил да използва такива алгоритми и да подобри постижението си на идващото състезание по пиене на минерална вода в дисциплината "Двойна серия".

При дисциплината "Двойна серия" много бутилки минерална вода, от разнообразни марки са наредени последователно на бара. Състезателят пие от последователни бутилки в редицата, като Журито записва марките минерална вода, които са били изпити. Това продължава дотогава, докато състезателят обяви, че е направил "серия". След това, според правилата, той трябва да повтори серията (т.е. да изпие още толкова от следващите в редицата бутилки така, че видовете им да образуват същата последователност). Оценяването се определя от големината на серията, времето за изпълнение и това дали състезателят е успял наистина да направи двойна серия.

Станчо държи рекорда на "Двойна серия", но иска да го подобри. Стратегията му трепач е специално оптимизирана – той тръгва от някакво място в редицата, и се движи надясно изпивайки всичко по пътя си. След определен брой бутилки обявава серия, и продължава със същия брой бутилки от мястото, където е завършил първата серия. За да бъде успешна стратегията, Станчо се нуждае от програма, която да избере мястото в редицата, от което да тръгне и броя бутилки които да образуват първата му серия. Поради специалните алгоритми за аритметика които ползва, е необходимо дължината на серията да дели броя на пропуснатите бутилки. Вие трябва да напишете тази програма вместо него.

Вход

На всеки ред на стандартния вход е зададен по един низ с дължина от 2 до 4000000, съставен от малки латински букви. Всяка буква означава една марка минерална вода.

Изход

За всеки низ програмата трябва да изведе на отделен ред на стандартния изход броя бутилки които трябва да пропусне Станчо от ляво надясно до началото на серията и дължината на серията, разделени с интервал, така че в резултат да се получи двойна серия с максимална дължина. Ако има повече от една максимална двойна серия, програмата трябва да посочи тази, която започва най-вляво. Винаги има поне една такава двойна серия във входните данни.

Вход	Изход
gagagaaabaabab kakaribaribarak	6 3 4 4

XIX РЕПУБЛИКАНСКА СТУДЕНТСКА ОЛИМПИАДА ПО ПРОГРАМИРАНЕ
ВАРНА, 4–5 МАЙ 2007

Задача J. 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 претенцията 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 ($0 \le a, b \le 2000$) 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 0 0, 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 6 3 0 0	1 2