ACM-ICPC Thailand Southern Programming Contest 2016
Hosted by
Songkhla Rajabhat University, Songkhla, Thailand

10 September 2016

Contest Problems

- There are 10 problems (A-J) to solve within 4 hours (240 minutes).
- Solve as many problems as you can, in an order of your choice.
- Use C or C++ or Java to program at your convenience for any problems.
- Input and output of each program are **standard input** and **standard output**.

Problem A	Pokémon Go
Problem B	Chromosome
Problem C	Count Prime Numbers
Problem D	Bird's Nest
Problem E	Start Up Project
Problem F	Modular Equation
Problem G	Plagiarism
Problem H	Sudoku Table Checking
Problem I	Simple Logical Operations
Problem J	Maximum Profit

Problem A. Pokémon Go

Time Limit: 2 sec

Typically, the player of PokemonGo can keep at most 250 pokemon. The new version of PokemonGo gives each player an initial sack of pokemons. So that, the player can select a set of pokemons to put in his/her sack. However, there is a limit on the weight that each sack can hold. Thus, the total weight of the pokemon selected must not exceed the limit of the sack. Furthermore, each player wants to select the pokemon with the high CP value. Therefore, the selection must maximize the total CP value as well.

Input

The first line of standard input contains an integer T, the number of test cases ($1 <= T <= 100$). T data sets follow. Each test case consists of four lines. The first line contains an integer N, the number of Pokemons where $1 <= N <= 50$. The second line contains an integer W, the capacity of the initial sack where $1 <= W <= 20$. The third line consists of N integers, each of which represents the weight of each Pokemon. The last line consists of N integers, each of which represents the CP value of each Pokemon. The weight of each Pokemon is ranged between 1 and 5 while the CP value of each Pokemon is ranged between 1 and 100.

Output

For each case, you are to output a line giving the maximum total CP value possible of that test case.

Sample Input

```
2
8
12
2 3 4 5 3 1 5 1
3 14 5 28 8 38 87 10
5
9
2 5 4 5 2
93 44 15 18 10
```

Sample Output

```
163
147
```

Problem B. Chromosome

Time Limit: 2 sec

At a biomedical research center, the technician must examine the string of chromosome for a particular pattern. Each string consists of any uppercase alphabet (i.e., A, B, C, .., Z). Currently, the technician must find the pattern that exactly matches the following rules:

Starting with zero or exactly one alphabet from {A, B, C, D, E, F}
Followed by at least one or more of alphabet A
Followed by at least one or more alphabet F
Followed by at least one or more alphabet C
Followed by zero or exactly one alphabet from {A, B, C, D, E, F}

You job is to write a program that performs similar task.

Input

You are given one string per line. Each line contains only uppercase alphabet without any whitespace and at most 200 alphabets. The input is terminated with an EOF (end-of-file / there is nothing left to read in).

Output

For each string/line, output "`Infected!`" in one line if that string matches the pattern described above. Otherwise, output "`Good`" in one line.
Note: Do not output the " symbol. It is used for clarification purpose.

Sample Input

```
AFC
AAFC
AAAFFCC
AAFCC
BAFC
QWEDFGHJMNB
DFAFCB
ABCDEFC
DADC
SDFGHJKLQWERTYU
AAAAAAAAAAAAABBBBBBBBBBBBBBCCCCCCCCCCCCCCCCCCDDDDDDDDDDDDEEEEEEEEEEEEEEEEFFFFFFFFC
AAAFFFFFBBBBCCCAAAFFFF
ABCDEFAAAFFFCCCABCDEF
AFCP
AAFFCPP
```

Sample Output

```
Infected!
Infected!
Infected!
Infected!
Infected!
Good
Good
Good
Good
Good
Good
Good
Good
Good
Good
```

Problem C. Count Prime Numbers

Time Limit: 1 sec

Write a program to count prime numbers which are less than or equal to a specific value *n*. For example, given *n* = 10, there are 4 prime numbers (2, 3, 5, 7), given *n* = 19, there are 8 prime numbers (2, 3, 5, 7, 11, 13, 17, 19).

Input

Each line of the input contains the integer *n* (1<= *n* <=100,000).

Output

Print out for each *n,* the number of prime numbers which are less than or equal to *n*.

Sample input	Sample output
10 100 1013 72 3	4 25 170 20 2

Problem D. Bird's Nest รังนก

Time Limit: 1 sec

ในประเทศไทยมีนกแอ่นกินรังเข้ามาทำรังในที่พักอาศัยของประชาชนเป็นจำนวนมาก จนเกิดเป็นธุรกิจบ้านรังนก และทำให้เกิดปริมาณรังนกจำนวนมาก ช่วงเวลาหนึ่งที่ผ่านมาเกิดภาวะราคารังนกตกต่ำ จนทำให้เกิดปัญหาขาดทุนของผู้ประกอบการรังนก จนถึงประมาณปี 2557 ได้เกิดการรวมตัวของผู้ประกอบการรังนกเป็นสมาคมและชมรมหลาย ๆ กลุ่ม และจัดงานประมูลรังนกขึ้นเพื่อหวังว่าราคาซื้อขายรังนกจะดีขึ้น ในการประมูลรังนกมีขั้นตอนคร่าว ๆ ดังนี้

1. ผู้ขายจะบรรจุรังนกใส่ถุง ส่งให้กับผู้จัดประมูล (สมาคม, ชมรม)
2. ผู้จัดประมูลนำถุงรังนกมาชั่งน้ำหนักและติดหมายเลขถุงรังนก
3. ผู้ประมูลลงทะเบียนเข้าร่วมประมูลเพื่อรับหมายเลขผู้ประมูล
4. ผู้จัดประมูลส่งถุงรังนกให้กับผู้ประมูลแต่ละคนเสนอราคาจนครบทุกคน
5. ผู้จัดประมูลสรุปราคาสูงสุดของแต่ละถุง

สมมุติว่า ในงานประมูลรังนกครั้งหนึ่ง **มีถุงรังนกที่ต้องประมูล 5 ถุง** ได้แก่ ถุงหมายเลข 1-5 และ **มีผู้ประมูล 3 ราย** ได้แก่ ผู้ประมูลหมายเลข 1- 3 การประมูลแต่ละถุงผู้ประมูลจะต้องใส่ข้อมูล 3 อย่าง ได้แก่ หมายเลขถุงรังนก หมายเลขผู้ประมูล และราคาต่อกิโลกรัม เรียกว่า รายการเสนอราคา เช่น

1 1 21500 หมายถึง ถุงรังนกหมายเลข 1 ผู้ประมูลหมายเลข 1 เสนอราคา 21,500 บาทต่อกิโลกรัม

2 1 25000 หมายถึง ถุงรังนกหมายเลข 2 ผู้ประมูลหมายเลข 1 เสนอราคา 25,000 บาทต่อกิโลกรัม

1 2 22000 หมายถึง ถุงรังนกหมายเลข 1 ผู้ประมูลหมายเลข 2 เสนอราคา 22,000 บาทต่อกิโลกรัม

จงเขียนโปรแกรมเพื่อหาผู้ประมูลและราคาเสนอสูงสุดของถุงรังนกแต่ละถุงโดยอ่านรายการเสนอราคาทีละรายการจนครบ *N* รายการ (*N* เป็นจำนวนเต็ม >=1) โดยมีเงื่อนไขว่า

1. หากเสนอราคาเท่ากัน รายการเสนอราคา ของผู้ประมูล ที่เข้ามาประมวลผลก่อน จะได้รังนกถุงนั้นไป
2. ผู้ประมูลสามารถเสนอราคาถุงเดิมอีกครั้งได้ แต่ต้องให้ราคาสูงกว่าเดิม ไม่เช่นนั้น ราคาดังกล่าวจะไม่ถูกนำมาพิจารณา เช่น ทำรายการ 1 2 22000 แล้ว ทำรายการ 1 2 22500 ได้
3. ถ้าหมายเลขถุงรังนก หรือหมายเลขผู้ประมูล ไม่ถูกต้อง หรือราคาเสนอไม่อยู่ในช่วงตั้งแต่ 1,000-100,000 บาท ให้แสดงผลลัพธ์ -9 และจบการทำงานทันที

Input ข้อมูลนำเข้า

บรรทัดแรกของอินพุต คือค่า N ซึ่งเป็นจำนวนเต็มบวก มีค่าไม่เกิน 100 ระบุรายการของการเสนอราคา และ มีข้อมูล *N* บรรทัด ตามมา แต่ละบรรทัดคือข้อมูลการเสนอราคา ประกอบด้วย จำนวนเต็มสามจำนวนคือ หมายเลขถุง (1-5) หมายเลขผู้ประมูล (1-3) และ ราคาที่เสนอ

Output ข้อมูลส่งออก

ผลลัพธ์ที่ต้องการ คือ หมายเลขถุงรังนก หมายเลขผู้ประมูลที่ให้ราคาสูงสุด และราคาสูงสุด เรียงลำดับตั้งแต่ถุงที่ 1 – 5 โดยถุงที่ไม่มีการเสนอราคาเลย จะระบุหมายเลขผู้ประมูลสูงสุด เป็น 0 และ ราคาสูงสุด เป็น 0 บาท เช่น 4 0 0

ตัวอย่าง

Sample Input	Sample Output
3 1 1 15000 1 2 11500 2 3 14500	1 1 15000 2 3 14500 3 0 0 4 0 0 5 0 0
12 1 1 21000 2 1 22500 3 1 20500 4 1 15000 5 1 16000 4 2 17000 3 2 20500 2 2 32500 1 2 30500 1 3 10500 2 3 10000 3 3 11000	1 2 30500 2 2 32500 3 1 20500 4 2 17000 5 1 16000

Problem E. Start Up Project (โปรเจคสำหรับ start up)

Time Limit: 2 sec

บริษัท Start up ได้ประมูลโปรเจคใหญ่มาหนึ่งโปรเจค แต่ไม่มีคนมาทำงาน จึงประกาศรับสมัคร Freelance ทางอินเตอร์เน็ต โดยแบ่งงานออกเป็น n ส่วน จนวันนี้มีคนผ่านการสอบสัมภาษณ์แล้ว จำนวน N คน Project Manager จึงประชุมแบ่งงานกัน โดยแบ่งงานเป็น N งาน โดยให้แต่ละคนบอกส่วนของงานที่สามารถทำได้ทั้งหมด โดยเรียงลำดับงานที่ถนัดที่สุด หลังจากนั้น Project Manager จึงจัดหน้าที่ให้แต่ละคน ว่าต้องทำงานส่วนไหน โดยหลักการแบ่งงานคือ

1) ให้แจ้งจำนวนงาน และส่วนงานที่ถนัด โดยให้แจ้งตามลำดับการสมัคร
2) กำหนดงานที่ถนัดที่สุดให้แก่คนที่สมัครก่อนคนแรก
3) แต่ถ้าถนัดเพียงงานเดียว จะได้รับการกำหนดงานนั้น
4) ทุกคนต้องได้งานทำ แต่ละคนจะได้รับงานเพียงส่วนเดียวเท่านั้น

Input ข้อมูลนำเข้า

บรรทัดแรก คือ จำนวนของงาน (N), $1 < N < 100$ งาน

บรรทัดอื่นๆ ตัวเลขตัวแรก คือจำนวนงานที่ถนัด ของคนที่ 1, 2, 3, ..., N ในขณะที่ตัวเลขอื่นๆ ในบรรทัดคือ หมายเลขงานที่ถนัด จำนวนงานที่ถนัดมีอย่างน้อย 1 งานเสมอ

จำนวนบรรทัดของข้อมูลนำเข้าเท่ากับจำนวนของงาน บวกหนึ่ง และข้อมูลนำเข้าเป็นข้อมูลที่จะสามารถจัดสรรงานให้กับทุกคนได้เสมอ

Output ข้อมูลส่งออก

หมายเลขงานที่ได้รับการจัดสรรสำหรับผู้ร่วมงานโดยเรียงตามลำดับพนักงาน

Sample Input	Sample Output
4 1 2 3 1 2 3 2 1 4 2 2 3	2 1 4 3

Problem F: Modular Equation

Time limit: 2 seconds

Mod (%) is arithmetic for finding a remainder of a divisor. **Mod** is widely used in the cryptography domain in order to make reversed operation in mathematics more complex and difficult especially in Public Key Cryptography world.

> **For example:**
> 4*6 **mod** 7 = ?, then we can easily find the answer, it is 3.
> If we reverse to solve 4*x **mod** 7 = 3, then the answer 'x' is 6.
> We can rewrite above equation as: 4*x ≡ 3 **mod 7** (≡ หมายถึง เท่ากันโดยเอกลักษณ์)

The question of this problem is to find ***x*** for given ***a, b, n*** such that ***ax ≡ b mod n*** where ***a*** and ***n*** are co-prime. Co-prime means only 1 or -1 can divide both of them e.g., a=10, n=3 or a=17, n=29, etc.

Input

The input consists of a set of lines with the integer numbers ***a, b*** and ***n***, separated with space ($0 < a, b$, and $n < 9 \times 10^9$) respectively. Note that ***b*** must be less than ***n*** always. ***a*** and ***n*** are co-prime.

Output

The output line consists of the minimum positive value of integer x regarding to ***ax ≡ b mod n***

Sample Input	Sample Output
10 2 3	2

Sample Input	Sample Output
7 4 5	2

Sample Input	Sample Output
3 4 5	3

Sample Input	Sample Output
2 1 3	2

Sample Input	Sample Output
17 3 29	7

Problem G. Plagiarism

Time Limit: 12 sec

Plagiarism means the practice of taking someone else's work or ideas and passing them off as one's own (definition given by Google Translate).

In academic, plagiarism is a very serious matter and is illegal. Although everyone knows that there still are many cases got caught in the past. Nevertheless, limitation of human efforts might not be able to catch all the cases. So *ACM*, i.e., Association of (non) Copying Matter, comes up with an idea to detect plagiarism with software. In order to do that, they gather programmers (you) together and hope that you can invent an efficient algorithm to detect plagiarism. Note that because false-positive result is unacceptable and because *ACM* doesn't want to fire all human detector, they want you to write a program to tell how likely a document is copied from somewhere else.

We define the problem as follows.

Two parameters `K` and `L` are given.

Let `A = a_1a_2a_3... a_K` and `B = b_1b_2b_3 ... b_K` be strings of length K.

We say that `A` and `B` are similar if there are at least `L` indexes where the characters of `A` and `B` are identical.

For example, let `K=4` and `L=2`, string `aabb` and `abcd` are not similar because only characters in the 1-st index are identical.

On the other hand, `aabb` and `abab` are similar because characters in the 1-st index are identical and characters in the 4-th index are identical.

Let string `C = c_1c_2c_3...c_n` be a given corpus string and let string `D = d_1d_2d_3...d_m` be a given document.

ACM want you to count pairs of indexes (`i <= n-k+1, j <= m-k+1`) such that the substring of length `K` of `A` starting at index `i` and the substring of length `K` of `B` starting at index `j` are similar. This number is an indicator to detect how likely that `D` is copied from `C`.

Input

The first line contains an integer *T*<=`10`, the number of test cases.

For each test, the first line contains two non-negative integer `K <= 5,000` and `L <= 1,000`.

The second and the third line contain corpus `C` and document `D` in respective order. The corpus document contains only small letters (a-z). The length of all given strings are at least 1 and at most 10,000.

Output

Output should contain *T* lines.

In line *i*, correct program should print "`Case #i: P`" where *P* is the number of similar pair of indexes as described above.

Sample Input

```
6
4 2
xxxxaabbxxxx
ababxx
2 2
aaaaabbbbb
```

```
abab
5 3
abcdefghxx
bbcceeffxx
2 0
aospdiopasdiopasdiopasdiopcmiopqwidpocqiwop
asiopdcimqwopcmidqwpocdimqwopdcimqwopcdqwimocpiqmw
3 1
qwecmoiqweopmiecopqwimcopqwiemcoppq
qpoweicmqwopemiqwiemcopimopimopwqicicmepmopimeopimcopiceopwieopcmqwiim
coepimeopwqiemoqwpciopwciecopqwicmwopecimqwopeiqwcie
6 2
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

Sample Output

```
Case #1: 12
Case #2: 2
Case #3: 5
Case #4: 2058
Case #5: 1173
Case #6: 0
```

Problem H. Sudoku Table Checking

Time Limit: 1 sec

Sudoku table is a 9x9-grid containing 1 digit in each cell. The 9x9 grid can be divided evenly into nine 3x3 subgrids. Every row, every column and every 3x3 subgrid of the Sudoku table must contain all digits from 1 to 9. So there is no duplicate numbers in any row or any column or any subgrid. Your task is to write a program to check whether or not a 9x9 table is a Sudoku table.

3	5	6	8	7	1	9	4	2
9	8	4	6	2	5	7	1	3
7	1	2	4	9	3	5	6	8
8	9	7	1	4	2	6	3	5
1	2	3	9	5	6	8	7	4
4	6	5	3	8	7	2	9	1
5	3	1	2	6	9	4	8	7
6	7	8	5	3	4	1	2	9
2	4	9	7	1	8	3	5	6

Figure 1. An example of a Sudoku table

Input

The first line contains the number n ($1<=n<=10$) which is the number of test cases i.e. the number of 9x9 tables to be checked. Each table is presented in 9 lines and each line contains only 9 numbers separated by a space. There is an empty line between each table.

Output

For each table, print "`Case X: Result`" where x is the number of test case, starting from 1 and *Result* is "**YES**" if it is a Sudoku table or "**NO**" otherwise.

Sample input	Sample output
3 3 5 6 8 7 1 9 4 2 9 8 4 6 2 5 7 1 3 7 1 2 4 9 3 5 6 8 8 9 7 1 4 2 6 3 5 1 2 3 9 5 6 8 7 4 4 6 5 3 8 7 2 9 1 5 3 1 2 6 9 4 8 7 6 7 8 5 3 4 1 2 9 2 4 9 7 1 8 3 5 6 9 1 2 3 4 5 6 7 8 8 9 1 2 3 4 5 6 7 7 8 9 1 2 3 4 5 6 6 7 8 9 1 2 3 4 5 5 6 7 8 9 1 2 3 4 4 5 6 7 8 9 1 2 3	Case 1: YES Case 2: NO Case 3: NO

```
3 4 5 6 7 8 9 1 2
2 3 4 5 6 7 8 9 1
1 2 3 4 5 6 7 8 9

1 5 6 8 7 1 9 4 2
2 8 4 6 2 5 7 1 3
3 1 2 4 9 3 5 6 8
4 9 7 1 4 2 6 3 5
5 2 3 9 5 6 8 7 4
6 6 5 3 8 7 2 9 1
7 3 1 2 6 9 4 8 7
8 7 8 5 3 4 1 2 9
9 4 9 7 1 8 3 5 6
```

Problem I. Simple Logical Operations

Time Limit: 1 sec

In electronic devices, a logic gate is a physical device implementing a Boolean function. It performs a logical operation on one or more logical inputs, and produces a single logical output. There are four basic logical operators that we have to implement.
Table 1 shows the true table logical NOT operator that is more commonly called an inverter. The NOT operator works with one input and produce an output which reverse an input. Table 2 shows the truth table for AND, OR and XOR operators

Table 1. NOT Operator

Input	Output
1	0
0	1

Table 2. Logical Operators with two input

Input		Output		
A	B	A AND B	A OR B	A XOR B
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

You have to write the program to perform the simple logical operations for the given test cases.

Input

The input will start with an integer T ($1 < T < 200$), the number of test cases. Each of the test cases starts with one word and followed by one or two integers. The word means type of logical operator. The following integer is the input for each operator. Only NOT operator that has one input. The others have two input.

Output

For each test case, produce one line of output. This line contains a string "`true`" or "`false`" (without the quotes), which denotes the results of the corresponding logical operation. Look at the output for sample input for details.
(Hint: The output "`true`" is equivalent to 1 and "`false`" is equivalent to 0)

Sample Input	Sample Output
7 OR 1 1 OR 0 0 AND 1 1 AND 1 0 NOT 0 XOR 0 0 NOT 1	true false true false true false false

Problem J. Maximum Profit

Time Limit: 1 sec

Textile and handicrafts of Koh Yor is one of the most product of Songkhla province. Kor Yor community usually receives a lot of orders from customers. Each order has a deadline and the community will obtain different profit. To maximize the profit, we have to consider two factors which are duration of deadline and the profit in each order.

You have to write the program to find the sequence of order (or job) that maximize the profit. We can select to perform only one job at a time and if there are jobs that can yield the same maximum profit, select the job by their order.

Input

The input will start with an integer ***T*** (1< ***T*** < 100), the number of test cases. Each of the test cases starts with one integer, ***M***, that indicates the numbers of the orders. Each of the orders starts with one word and then two integers. The word means job or order name that can be up to 100 orders, starting from `J1,J2,J3` … The first integer is the deadline of this order (for example the number of days from now) and the second integer is the profit we will get from this job.

Output

For each line of input produce one line of output. Each line shows the sequence of order that we accept to produce and deliver. And we have to add the maximum profit we get at the end of the last sequence. Each order and maximum profit are separated by a colon `:` .

Sample Input	Sample Output
2 4 J1 3 200 J2 1 400 J3 1 100 J4 1 300 5 J1 2 1000 J2 1 200 J3 2 900 J4 3 100 J5 1 200	J2:J1:600 J2:J1:J4:1300

Explanation

From the sample input, there are two test cases. The first test case start with integer number,4, means that we have to consider four orders. In the next line `J1` is the name of order/job, integer 3 means duration of deadline and 200 is the profit of job `J1`.