University of Sharjah
College of Science
Department of Computer Science

University Programming Contest

November 12, 2017

Problem Set

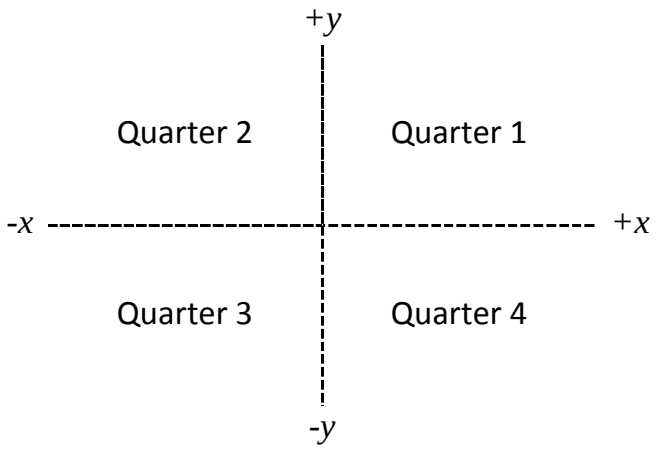
Problem	Balloon
Problem A: Quarter of a Point	Gold
Problem B: Social Worker	Blue
Problem C: Calm Numbers	Yellow
Problem D: Genome Sequencing	Green
Problem E: CS is Hiring	Red
Problem F: Electronic Waste	Pink
Problem G: Robopal	Black
Problem H: Perfect Contest	Purple
Problem I: Part-time Job	Orange
Problem J: Divisors	Silver

Do not open before the contest starts

Problem A: Quarter of a Point

Code file: quarter.{java|cpp}
Input file: quarter.in
Balloon color: Gold

The *xy*-coordinate plane is divided into four quarters as shown in the below figure. For any given point in the coordinates system, you need to find the quarter of the point. If the point lies on one of the axes then display "No quarter".



Input format:

Each input line contains the x and y coordinates of a point separated by a single space ($-10^8 \leq x, y \leq 10^8$). The input is terminated by x and y both equal to zero.

Output format:

For each point in the input, output a single line that displays "**`Quarter X`**" where **X** = 1,2,3 or 4 is the quarter of the point or "**`No quarter`**" if the point is on the x or y axis. No output for the input termination values (i.e. 0 0)

Sample input/output:

quarter.in

```
2 3
-1023998 13
45522 0
-900 -1
0 -12345
399 399
0 0
```

Output

```
Quarter 1
Quarter 2
No quarter
Quarter 3
No quarter
Quarter 1
```

Problem B: Social Worker

Code file: social.{java|cpp}
Input file: social.in
Balloon color: Blue

Mr. Shafeeq is a very dedicated social worker (*BTY, Shafeeq means kindhearted in Arabic*). He always works on creative ways to serve the community. Recently, Shafeeq has volunteered in a mission to supply milk to a childcare center. In this task, Shafeeq recycles empty milk bottles to buy new milk bottles. He gets some empty bottles from friends and some from his home. Besides, he recycles the consumed bottles from the childcare center. The childcare center immediately consumes all the milk supplied. And Shafeeq, repeatedly, recycles all the empty bottles he can every day.

Now, it is your turn to serve the community by calculating the total milk bottles Shafeeq can supply in a day.

Example:

Assume 3 empty bottles can be exchanged for 1 new bottle. Today, Shafeeq got 7 empty bottles from friends and 4 empty bottles from home. He can exchange 9 of the total 11 empty bottles with 3 new bottles and keep the 2 extra empty bottles. The 3 new bottles will be consumed by the childcare center and recycled. Shafeeq now has 5 empty bottles. He can exchange 3 of them with 1 new bottle. When this new bottle is recycled, Shafeeq can exchange his total 3 empty bottles with 1 new bottle. He will not keep any extra empty bottles. When the final new bottle is recycled, he cannot get new bottles with 1 empty bottle. The total milk bottles Shafeeq can supply in the day is 3+1+1 = 5.

Input format:

Every line of input is a separate case. The case contains three integers F, H and C ($0 \le F, H \le 1000; 2 \le C \le 2000$) where F and H are the number of bottles received from friends and home respectively, and C is the number of empty bottles required to buy one new milk bottles. The input is terminated with a line where all three values are 0.

Output format:

For each input case, output one line showing the number of new milk bottles that can be supplied for the case.

Sample input/output:

social.in

```
7 4 3
5 5 2
0 0 0
```

Output

```
5
9
```

Problem C: Calm Numbers

Code file: calm.{java|cpp}
Input file: calm.in
Balloon color: Yellow

A calm number is a number in which every pair of adjacent digits differ by at most 1. 54556 is a calm number where 134 is not. The sequence of digits 5, 4, 5, 5, 6 differs by 1 or less in every step while the sequence 1, 3, 4 differs by 2 between 1 and 3. Leading zeroes are not considered as part of the number (i.e. 054556 is still calm).

Your task is to classify a list of numbers as calm or not calm.

Input format:

Each input line contains one test case. A test case is a number N, where $1 \leq N \leq 10^{100}$. The input is terminated by $N = 0$.

Output format:

For each input case, the output should be a single line in the form "**k. ans**" where **k** is the case number and **ans** is either "Yes" or "No" (i.e. calm or not).

Sample input/output:

calm.in

```
54556
134
054556
545560
98776655443456655432101 22
98776655443456655432101 20
12345678986543210012
0
```

Output

```
1. Yes
2. No
3. Yes
4. No
5. Yes
6. No
7. No
```

Problem D: Genome Sequencing

Code file: genome.{java|cpp}
Input file: genome.in
Balloon color: Green

The study of DNA structures of living organisms is a special field of intersection between biology and computer science. The DNA structure is decoded as a string that uses an alphabet of four letters: 'A', 'G', 'T' and 'C'. One special interesting query is the presence (or absence) of certain connected sequences (substrings) within the genome string.

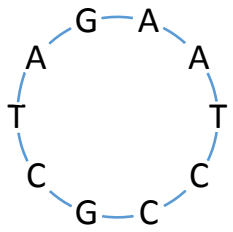
Your friend Rabiant, a biotech student, is doing a research on circular bacterial genome. She was able to decode the genome of the species under study and she has some special substrings to test their presence.

The genome string is circular (i.e. the two ends of the string are connected). While searching for substrings, direction is insignificant. The existence of the connected sequence moving in either direction should be counted.

Your task is to help Rabiant by reporting whether the connected sequences are found within the circular genome.

Example:

The genome string "AATCCGCTAG" represents the below circular structure:



Substrings to search within the genome string are:

GCTA present AATCCGCTAG

GCC present AATCCGCTAG

TAAGA present AATCCGCTAG

TCGA not present AATCCGCTAG

Input format:

The input starts with a number T ($1 \leq T \leq 100$) that represents the number of test cases in the file. Each case contains 5 lines. The first line is a string G that represents the circular genome ($2 \leq length(G) \leq 255$). The following 4 lines contain 4 substring S_i ($i = 1, 2, 3$ and 4; $2 \leq length(S_i) \leq 50$) each in a separate line to test their presence in the circular genome G.

Output format:

For each input case, output a 4-bits binary number in a separate line with each bit set to 1 if the corresponding substring S_i is present in G and to 0 otherwise. S_1 corresponds to the leftmost bit and S_4 to the rightmost bit.

Sample input/output:

genome.in

```
2
AATCCGCTAG
GCTA
GCC
TAAGA
TCGA
CCTAAG
GA
TG
TA
AC
```

Output

```
1110
1010
```

Explanation of sample input/output:

The input contains 2 test cases. In the first test case the circular genome string G is AATCCGCTAG (given in the second line of input). The 4 substrings to test are given in the following 4 lines. The results of testing their presence in G in order is: found, found, found and not found. Thus the output is 1110. The second case follows and starts with the circular genome string CCTAAG.

Problem E: CS is Hiring

Code file: hiring.{java|cpp}
Input file: hiring.in
Balloon color: Red

The Department of Computer Science is hiring a new member. The department received huge number of applications for the position, and now the head of department, Prof. Zaher, is so busy arranging and ranking these applications to make his recommendations. Luckily, the applications are organized in a table that contains all the parameters required for ranking the applicants. Your task is to help Prof. Zaher find the highest 3 applicants based on different ranking criteria.

Example:

Given the below data table (6 rows and 5 columns):

1	Khalid	8	A1	265	A0087622
2	Ali	10	A2	325	A1988266
3	Yahya	15	D3	375	T5429360
4	Rawda	7	B1	280	W0882151
5	Basel	3	B1	400	T9366426
6	Sham	22	B1	385	F7330425

Columns 1, 3 and 5 are text while columns 2 and 4 are numeric (the serial number is not a column).

Possible queries on the data table:

Query column	**Result**	**Explanation**
2, Descending	6 3 2	The highest values in column 2 are 22, 15 and 10
4, Ascending	1 4 2	The lowest values in column 4 are 265, 280 and 325
3, Ascending	1 2 4	The lowest values in column 2 are A1, A2 and B1. The tie on B1 is resolved by selecting the lower index.

Input format:

Each test case starts with a line that contains 3 values: Number of columns in the table C ($1 \leq C \leq 10$), number of rows in the table R ($3 \leq R \leq 100$), and number of queries Q ($1 \leq Q \leq 20$). The second line in the test case contain C integers (1 or 2) that specify the data type of each column in the table: 1 is text column and 2 is integer column. R lines follow for each row in the table with C values separated by a single space in each line. Q lines follow for each query. The query has a format of "**X B**" where **X** is the index of the column used in ranking and **B** is either 'A' or 'D' representing ascending or descending ordering respectively.

A test case with C, R and Q all set to 0 indicates the end of input file.

Output format:

For each query, output one line with the 3 integers separated by a single space. The 3 integers are the indices of top 3 rows based on the query. If 2 rows have the same ranking, the row with lower index should appear first.

Sample input/output:

hiring.in

```
5 6 3
1 2 1 2 1
Khalid  8  A1  265  A0087622
Ali  10  A2  325  A1988266
Yahya  15  D3  375  T5429360
Rawda  7  B1  280  W0882151
Basel  3  B1  400  T9366426
Sham  22  B1  385  F7330425
2 D
4 A
3 A
0 0 0
```

Output

```
6 3 2
1 4 2
1 2 4
```

Problem F: Electronic Waste

Code file: ewaste.{java|cpp}
Input file: ewaste.in
Balloon color: Pink

Recycling electronic waste is a growing trend to reduce one of the serious disadvantages of digital technology. UoS has appointed you to recycle its electronic waste. Different electronic pieces in the waste are listed by their weights in grams. The recycling process you are planning to use depends on the range of weights of all pieces (i.e. the difference between heaviest and lightest pieces). You need to calculate this range.

Input format:

The input starts with a number T ($1 \le T \le 1,000$) that represents the number of test cases in the file. Each case starts with the number of pieces in the case N ($2 \le N \le 10^6$). N values that represent the weights W_i ($1 \le W_i \le 10^9$; $1 \le i \le N$) of pieces follow.

Output format:

For each input case, the output should be a single line "**`The range is X`**", where **X** is the calculated range of weights of pieces in the test case.

Sample input/output:

ewaste.in

```
2
4
1
1
100000000
1
5
20
2
20005
200
2000
```

Output

```
The range is 99999999
The range is 20003
```

Explanation of sample input/output:

The input contains 2 test cases. The first test case contains 4 pieces with weights 1, 1, 100000000 and 1. The range is 100000000 – 1 = 99999999. The second test case contains 5 pieces with weights 20, 2, 20005, 200 and 2000. The range is 20005 – 2 = 20003.

Problem G: Robopal

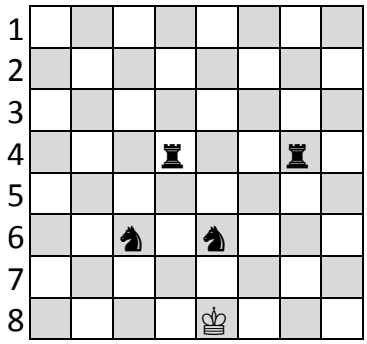
Code file: robopal.{java|cpp}
Input file: robopal.in
Balloon color: Black

Your robot friend Robopal is playing chess. He is well trained to self-adapt and understand different situations, but in some circumstances he may need your help (*that's what friends are for*). Now, Robopal's King is under attack. The opponent player is using knights and rooks only in the attack. Robopal needs your help to find the safe squares where he can move his King.

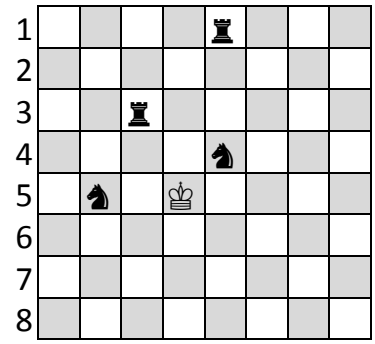
In chess, the rook moves and attacks in straight lines in any of the four directions. The rook cannot move beyond a square used by another piece. The knight jumps to, and attacks, the squares at L or inverted L positions. There are at most 2 knights and 2 rooks. The king can move to any adjacent square (side or diagonal). A square is considered safe if Robopal's King can capture the piece in that position without being threatened.

Sample settings:

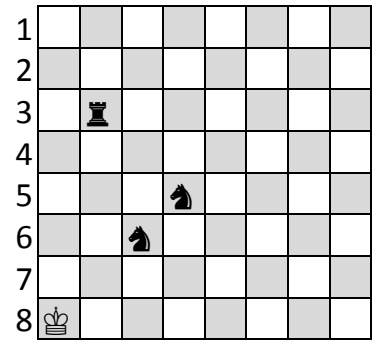
Robopal is always the white player.



Safe squares: **f7**.



Safe squares: **e5** and **e6**.



Safe squares: **none**.

Input format:

The input starts with a number T ($1 \le T \le 1,000$) that represents the number of test cases in the file. Each case starts with the number of pieces in the case N ($2 \le N \le 5$). N lines follow in the format **xn P**, where **xn** is the cell location of the piece using the above column/row code and **P** is either K (for the white king), R (for a black rook) or N (for a black knight). In each case, there will be exactly 1 K and at most 2 R's and 2 N's.

Output format:

For each input case, the output should be a single line in the form "**k. ans**" where **k** is the case number and **ans** is the number of safe squares around the king.

Sample input/output:

The below robopal.in file shows the 3 sample settings provided above.

robopal.in

```
3
5
e8 K
e6 N
c6 N
g4 R
d4 R
5
e1 R
c3 R
e4 N
b5 N
d5 K
4
d5 N
a8 K
c6 N
b3 R
```

Output

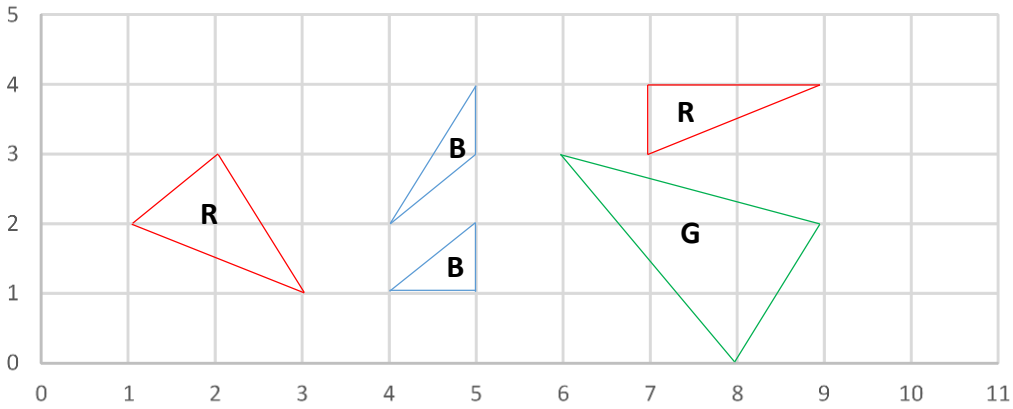
```
1. 1
2. 2
3. 0
```

Problem H: Perfect Contest

Code file: perfect.{java|cpp}
Input file: perfect.in
Balloon color: Purple

UPC, University of Perfect Coders, is organizing its annual programming contest. All computer labs in the university are perfect triangular rooms. These computer labs belong to different colleges in the university. Your task is to select a lab to run the contest. You want the maximum area lab. The labs supervisor, being so perfect also, imposed one condition. You cannot use a lab from a college that owns only one lab. Find the maximum area you can use for your contest.

Example:



In the above figure, 5 labs belong to 3 departments R, G and B. The labs are located at:

Lab 1: (2,3), (1,2) and (3,1) belongs to R.

Lab 2: (4,1), (5,1) and (5,2) belongs to B.

Lab 3: (4,2), (5,3) and (5,4) belongs to B.

Lab 4: (8,0), (6,3) and (9,2) belongs to G.

Lab 5: (7,3), (7,4) and (9,4) belongs to R.

Lab 4, which belongs to college G, cannot be used (as G has only 1 lab). The largest lab from the remaining labs is Lab 1 with area of 1.5 square units.

Input format:

The input starts with a number T ($1 \leq T \leq 1000$) that represents the number of test cases in the file. Each case contains a set of labs. Each lab is displayed in a separate line in the form: $A\ X1\ Y1\ X2\ Y2\ X3\ Y3$ where A is an uppercase letter that represents the college that owns the lab. $X1$, $Y1$, $X2$, $Y2$, $X3$ and $Y3$ (all integers) are the coordinates of the vertices of the triangle ($-1000 \leq X1, Y1, X2, Y2, X3, Y3 \leq 1000$). The test case ends with a line where A is a dash '-'. The maximum number of labs in a test case is 200.

Output format:

For each test case in the input, output 1 line that contains the area of the selected lab rounded to 1 decimal place.

Sample input/output:

perfect.in

```
3
R 2 3 1 2 3 1
B 4 1 5 1 5 2
B 4 2 5 3 5 4
G 8 0 6 3 9 2
R 7 3 7 4 9 4
-
V 8 -2 -9 -10 -8 -4
M -3 3 0 8 -3 10
B 8 0 -4 -7 1 -4
I 3 -5 -1 -3 -8 -9
O -10 1 4 1 4 5
-
A -59 -19 -47 8 33 19
A 80 85 -40 -81 19 -51
A 29 -28 -37 42 -88 -50
B 100 35 50 -39 15 -68
B 63 -85 97 -96 -24 -86
B 18 -100 31 45 90 -63
-
```

Output

```
1.5
0.0
4979.5
```

Problem I: Part-time Job

Code file: parttime.{java|cpp}
Input file: parttime.in
Balloon color: Orange

It's time to make some money using your programming skills (*apart from the contest prize* ☺). You got a part-time programming job. Your employer offers daily tasks. As a part-timer, you may accept or decline any task but, because you have your student's duties, you cannot work two consecutive days (i.e. with no break). The daily tasks have different pays and, naturally, you want to maximize your total pay. Given the payments offered for the tasks in a certain period, find the maximum amount of money you can make.

Example:

Given the pay for 7 days tasks as: 175, 350, 150, 150, 275, 350 and 250. The maximum total pay is 875 by accepting tasks 2, 5 and 7 (350+275+250).

Input format:

The input consists of multiple cases. Each case is a list of tasks pays P_i ($1 \le i \le 100$; $1 \le P_i \le 1,000$). Each case is terminated by a value of 0. A case that starts with 0 indicates the end of the input file.

Output format:

For each input case, output a single line in the form "**k. ans**" where **k** is the case number and **ans** is the maximum possible total pay.

Sample input/output:

parttime.in

```
175 350 150 150 275 350 250 0
100 200 100 200 100 0
656 265 512 855 512 661 645 497 0
0
```

Output

```
1. 875
2. 400
3. 2669
```

Problem J: Divisors

Code file: divisors.{java|cpp}
Input file: divisors.in
Balloon color: Silver

Prime factorization is the decomposition of an integer into a product of prime numbers. For example, the prime factorization of 24 is $2x2x2x3$. This expression is usually compacted using power notation by representing the number as a product of powers of its prime factors. This makes 2^3x3 the prime factorization of 24. It is a well-established mathematical fact that every positive integer has a unique prime factorization.

Your task is to read an arbitrarily large positive integer N represented using prime factorization and check whether a set of numbers are divisors of N or not.

Examples:

Given $N = 2^3x3^2x5$, the following numbers are classified as divisors of N (or not):

10	3	25	18	45	49	400	40
Yes	Yes	No	Yes	Yes	No	No	Yes

Given $N = 2x37^3x61^5x67^{12}x229^6$, the following numbers are classified:

10	25	15343	3513547	50653	1874161	40401	2479
No	No	Yes	Yes	Yes	No	No	Yes

Input format:

The input file starts with an integer C ($1 \le C \le 1,000$) that represents the number of test cases. Each test case start with the number of prime factors P in the prime factorization of N ($1 \le P \le 100$) followed in the same line by P pairs of values separated by single space where each pair p_i e_i is one term $p_i^{e_i}$ in the prime factorization of N ($1 \le p_i, e_i \le 1000$) and p_i is prime. The second line in each test case is the number of values D ($1 \le D \le 100$) to test for division followed by D lines each contains 1 integer d_i ($1 < d_i \le 10^9$).

Output format:

For each value d_i display either "Yes" or "No" in a separate line specifying whether N divides d_i or not.

Sample input/output:

divisors.in

```
2
3 2 3 3 2 5 1
8
10
3
25
18
45
49
400
40
5 2 1 37 3 61 5 67 12 229 6
8
10
25
15343
3513547
50653
1874161
40401
2479
```

Output

```
Yes
Yes
No
Yes
Yes
No
No
Yes
No
No
Yes
Yes
Yes
No
No
Yes
```

Explanation of sample input/output:

The input contains 2 test cases. The first test case N has 3 terms in the prime factorization $2^3 x 3^2 x 5^1$. 8 values should be tested as divisors of N: 10, 3, 25, …, 40. For every value there is one output line of “Yes” or “No”. The second test case N contains 5 terms in the prime factorization…