Problem A

Automat

(2 sec)

Input: a.in
Output: standard output

Consider a device doing some work. Besides, it must control its temperature. It must be neither too high nor too low. But something wrong happens with the temperature controlling algorithms, and device acts as follows:
Each minute, it selects one of possible actions changing its temperatures and executes it. Each action is selected with the given probability based on current temperature.
Also, each minute temperature goes **1** degree below.
You are given the parameters of temperature changes, count the probability that the temperature stays in the given range.

Input:
First line of input contains the quantity of tests **T** (**1≤T≤20**).
First line for each test contains numbers **A**, **B**, **C**, **N**, where **A** is the minimal allowed temperature, **B** is the maximal allowed temperature, **C** is the starting temperature, **N** is the number of minutes to work. **0≤A≤B≤30**, **A≤C≤B**, **0≤N≤30**.
Each of the next **B–A+1** lines contains **7** nonnegative integers that sum up to **100** – percentages of changes **–3, –2, –1, 0, 1, 2** and **3** degrees correspondingly.

Output:
Output **T** lines of the form “**Case #A: B**”, where **A** is the number of test (beginning from **1**), **B** is the desired probability this test case.

Input	Output
2 1 2 1 2 0 0 0 10 50 40 0 0 0 50 0 0 30 20 3 5 4 5 0 0 0 10 20 30 40 0 0 10 20 30 40 0 0 10 20 30 40 0 0	Case #1: 0.45 Case #2: 0.59049

All-Ukrainian Collegiate Programming Contest (AUCPC)

AUCPC Finals

- 2010
- 2009
- 2008
  - Schedule
  - Standing
  - Problems
  - Problem A
  - Problem B
  - Problem C
  - Problem D
  - Problem E
  - Problem F
  - Problem G
  - Problem H
  - Problem I
  - Problem J
  - Sponsors
  - Photos
- 2007
- 2006
- 2005

AUCPC Sub-Contests

Site map

AUCPC Finals -> 2008 -> Problems

Problem B

BELL NUMBERS
(2 sec)

Input: b.in
Output: standard output

The Bell numbers B_n describe the number of ways a set with n elements can be partitioned into disjoint, non-empty subsets. For example, $B_0 = 1$ because we have only one partition of empty set. $B_3 = 5$ because we have **5** different partitions of the set {*a*, *b*, *c*}: {{**a**}, {**b**}, {**c**}}, {{**a**, **b**}, {**c**}}, {{**a**, **c**}, {**b**}}, {{**a**}, {**b**, **c**}}, {{**a**, **b**, **c**}}.
The determinant D_n is given:

$$D_n = \begin{vmatrix} B_0 & B_1 & B_2 & \dots & B_n \\ B_1 & B_2 & B_3 & \dots & B_{n+1} \\ \dots & \dots & \dots & \dots & \dots \\ B_n & B_{n+1} & B_{n+2} & \dots & B_{2n} \end{vmatrix}$$

The prime number p is given. Find the greatest integer k, for which p^k divides D_n.

Input. Each line has two integers: n and p (**0** n, p **10000**). It is known that p is prime.

Output. For each line of input print on a separate line the greatest integer k, for which p^k divides D_n.

Input	Output
1 5 3 2 4 2 4 3 10000	0 2 5 2 24962375

Developed by Deep Vision

Всеукраинская студенческая олимпиада
по программированию

Problem C

Longest chain

(4 sec)

Input: c.in
Output: standard output

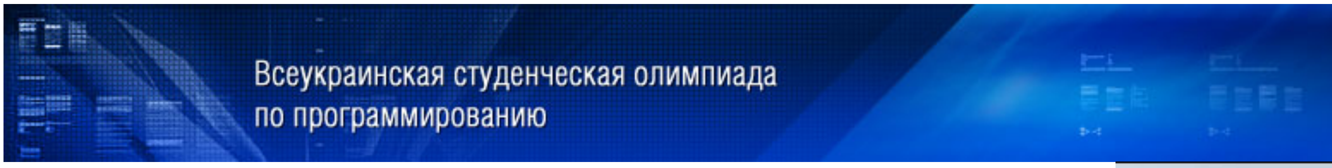
You are given a list of positive integers. Find the number of elements in the largest subset that can be arranged in a chain so that from each two neighboring elements one of them divides another one.

Input:
First line of input contains the quantity of tests **T** (**1≤T≤35**).
Each of the next **T** lines contain number of elements **N** (**1≤N≤17**) and **N** positive integers, each number will be between **1** and 10^9 inclusive.

Output:
Output **T** lines of the form “**Case #A: B**”, where **A** is the number of test (beginning from **1**), **B** is the desired number for this test case.

Input	Output
2 3 1 2 3 5 2 3 4 5 6	Case #1: 3 Case #2: 4

Developed by Deep Vision



РУС | УКР | ENG

All-Ukrainian Collegiate Programming Contest (AUCPC)

AUCPC Finals

- 2010
- 2009
- 2008
  - Schedule
  - Standing
  - Problems
    - Problem A
    - Problem B
    - Problem C
    - Problem D
    - Problem E
    - Problem F
    - Problem G
    - Problem H
    - Problem I
    - Problem J
  - Sponsors
  - Photos
- 2007
- 2006
- 2005

AUCPC Sub-Contests

Site map

AUCPC Finals -> 2008 -> Problems

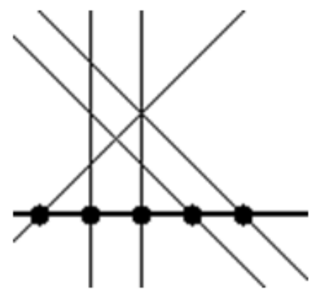
Problem D

Lines

(4 sec)

Input: d.in
Output: standard output

You are given a positive integer **N**. Let **A, B** and **C** be non-negative integers, such **that** $A+B+C=N$. Let there be **N** marked points on a line with an equal distance between neighboring ones. Draw lines at an angle of **45** degrees through the leftmost **A** points, draw lines at an angle of **90** degrees through the next **B** points, and at an angle of **135** degrees – through the last **C** points. These lines will intersect in some of points.



For clarity see the image, where **N=5**, **A=1**, **B=2**, **C=2**. There are **6** points of intersection.

Your task is quite simple – for given **N** you are to count the sum of quantities of intersection points for all possible triples **A, B, C**.

Input:
First line of input contains the quantity of tests **T** ($1 \le T \le 1000$).

Each of the next **T** lines contains an integer **N** ($2 \le N \le 10^6$) – the quantity of points on the line in a current test.

Output:
Output **T** lines of the form "**Case #A: B**", where **A** is the number of test (beginning from **1**), **B** is the sum of quantities of intersection points for given **N**.

Input	Output
3 2 3 5	Case #1: 3 Case #2: 13 Case #3: 91

Developed by Deep Vision

Problem E

Pareto's domination

(25 sec)

Input: e.in
Output: standard output

A point with coordinates (x_1, x_2, ..., x_n) is called dominated in Pareto's sense by a point with coordinates (y_1, y_2, ..., y_n), if for each i ($1 \le i \le n$) the inequality $x_i \le y_i$ holds. A set of some points is given. Your task is to find the number of points in this set that are not dominated in Pareto's sense by any other point in the given set.

Input:

First line of input contains the quantity of tests **T** (**1≤T≤10**).
First line of each test case contains two numbers: **N** (**1≤N≤50000**) – the number of points in the set and **M** (**1≤M≤4**) – the space dimension. Then there are **N** lines, each of which contains **M** integers – coordinates of a point, separated by spaces (each coordinate is less than 10^9 by its absolute value). All points in the set are different.

Output:

Output **T** lines of the form "**Case #A: B**", where **A** is the number of test (beginning from **1**), **B** is the quantity of non-dominated points.

Input	Output
2 **4 1** **1** **2** **3** **4** **4 2** **0 0** **1 1** **2 0** **0 2**	**Case #1: 1** **Case #2: 3**

Developed by Deep Vision

Problem F

Permutations

(2 sec)

Input: f.in
Output: standard output

Butt-head has n volumes of his favourite Encyclopedia For Stupid Teenagers. The volumes are numbered from **1** to n and arranged in a row. He doesn't like strict order, but he dislikes complete chaos as well. Butt-head counts the ***distance*** of volumes' pertmutation as the sum of differences between number and position for all volumes. In other words, if the permutation is $(i_1, i_2, \ldots i_n)$, where i_k ($1 \le k \le n$) denotes the number of volume on k-th place, then its distance will be $|i_1-1|+|i_2-2|+\ldots+|i_n-n|$. Butt-head has a favorite number d, and he wants to arrange the Encyclopedia volumes so, that the distance will be equal to d. In how many ways can he do this?

Input:
First line of input contains the quantity of tests **T** ($1 \le T \le 100$).
Each of the next **T** lines contains data for one test: the quantity of volumes n ($1 \le n \le 50$) and the required distance d ($0 \le d \le 10000$), separated by a space. Numbers n, d are integer.

Output:
Output **T** lines of the form "**Case #A: B**", where **A** is the number of test (beginning from **1**), **B** is the quantity of permutations of n volumes with distance d, taken modulo **100007**.

Input	Output
5 2 0 2 2 4 1 4 2 4 6	Case #1: 1 Case #2: 1 Case #3: 0 Case #4: 3 Case #5: 9

Всеукраинская студенческая олимпиада
по программированию

РУС | УКР | ENG

AUCPC Finals -> 2008 -> Problems

Problem G

Polyhedron

(4 sec)

Input: g.in
Output: standard output

You are given a set of points in 3-D space. Your task is to calculate the number of ***k***-vertex sides of the convex polyhedron with minimal volume that contains the whole given set of points.

Input:
First line of input contains the quantity of tests **T** (**1≤T≤100**).
First line of each test case contains the number of points in given set **N** (**4≤N≤30**).
Then N lines follow, each of which contains **3** integer numbers: **X, Y, Z (−1000≤X, Y, Z≤1000**) – coordinates of points. It's guaranteed that any convex polyhedron, that contains all given points, has a positive volume.

Output:
For each of **T** test cases output a line of the form "**Case #A:**", where **A** is the number of test (beginning from **1**), and then – **M** more lines, where **M** is the quantity of different side types (a side type means quantity of its vertices). In each of the next **M** lines you have to output two numbers: ***k*** – the number of vertices in the side and q_k – the number of ***k***-vertex sides in the polyhedron. After ***k*** you must print a colon "**:**" and then – a space. You have to output the result in ascending order of ***k***.

Input	Output
2 6 0 0 0 2 0 0 0 2 0 2 2 0 3 1 0 1 1 1 5 0 0 0 1 1 0 0 2 0 2 1 0 1 1 1	Case #1: 3: 5 5: 1 Case #2: 3: 4

Developed by Deep Vision

Всеукраинская студенческая олимпиада по программированию

РУС | УКР | ENG

All-Ukrainian Collegiate Programming Contest (AUCPC)

AUCPC Finals

- 2010
- 2009
- 2008
  - Schedule
  - Standing
  - Problems
  - Problem A
  - Problem B
  - Problem C
  - Problem D
  - Problem E
  - Problem F
  - Problem G
  - Problem H
  - Problem I
  - Problem J
  - Sponsors
  - Photos
- 2007
- 2006
- 2005

AUCPC Sub-Contests

Site map

AUCPC Finals -> 2008 -> Problems

Problem H

Rope pulling – 2

(2 sec)

Input: h.in
Output: standard output

Maybe, you heard the story about programmers, who pulled the rope. They wanted to reach the situation, when each programmer had competed with each other. They desire to pull the rope again. But now they want, that each pair of programmers will stand in the opposite teams at least *twice*.
Number of programmers is even. They hold some rounds, and in each round all programmers divide into **2** *equal* teams. Programmers are very lazy :), so they want to hold as few rounds as possible. What is the minimal quantity of rounds, for which exist such schedule, that any two programmers will be in opposite teams in at least two rounds?
For example, if we have **8** programmers, numbered from **1** to **8**, then we can organize the division in the following way:
(1, 2, 3, 4) – (5, 6, 7, 8);
(1, 2, 5, 6) – (3, 4, 7, 8);
(1, 3, 5, 7) – (2, 4, 6, 8);
(1, 4, 6, 7) – (2, 3, 5, 8).

Input:
First line of input contains the quantity of tests **T** (**$1 \le T \le 50$**).
Each of the next **T** lines contains an even integer **N** (**$2 \le N \le 300$**) – the quantity of programmers in a current test.

Output:
Output **T** lines of the form "**Case #A: B**", where **A** is the number of test (beginning from **1**), **B** is the needed quantity of rounds for given **N**..

Input	Output
4 2 4 12 16	Case #1: 2 Case #2: 3 Case #3: 5 Case #4: 5

Developed by Deep Vision

Problem I

Sets

(25 sec)

Input: i.in
Output: standard output

There is a graph with vertices divided into **4** groups, and edges join only vertices from **1** and **2, 2** and **3, 3** and **4,** and **4** and **1** groups. You have to count the maximal number of *non-intersecting* **4**-vertice sets, such that each set contains **1** vertex from each group, and they are connected in a cycle (vertex from group **1** is connected with vertex from group **2**, vertex from group **2** is connected with vertex from group **3**, vertex from group **3** is connected with vertex from group **4**, vertex from group **4** is connected with vertex from group **1**).

Input:
First line of input contains the quantity of tests **T** ($1 \le T \le 10$).
Each test case begins with the line containing **4** numbers: N_1, N_2, N_3, N_4 – number of vertices in groups **1, 2, 3** and **4** ($1 \le N_1, N_2, N_4 \le 10$, $1 \le N_3 \le 7$).
Then $2N_1$ lines follow, $2i$-th line contains the number of vertices from group **2**, i-th vertex from group **1** is connected with, and then numbers of these vertices, $(2i+1)$-st line contains the number of vertices from group **4**, i-th vertex from group **1** is connected with, and then – numbers of these vertices (vertices in each group are numbered from **0** to N_P-1, **P** is number of group).
Then $2N_3$ lines follow, $2i$-th line contains the number of vertices from group **2**, i-th vertex from group **3** is connected with, and then numbers of these vertices, $(2i+1)$-st line contains the number of vertices from group **4**, i-th vertex from group **3** is connected with, and then – numbers of these vertices.

Output:
Output **T** lines of the form "**Case #A: B**", where **A** is the number of test (beginning from **1**), **B** is the desired number for this test case.

Input	Output

`2`
`1 3 2 4`
`3 0 1 2`
`4 0 1 2 3`
`2 0 1`
`1 0`
`2 1 2`
`2 1 3`
`4 7 4 7`
`3 0 1 5`
`3 0 1 5`
`3 0 2 6`
	`Case #1: 1`
	`Case #2: 3`

3 0 2 6 3 1 2 6 3 1 2 6 3 0 1 2 3 0 1 2 2 0 2 2 0 2 1 0 1 0 2 1 0 2 1 0 2 0 1 2 0 1	

Developed by Deep Vision

Всеукраинская студенческая олимпиада по программированию

РУС | УКР | ENG

Problem J

Tokens

(4 sec)

Input: j.in
Output: standard output

You are given a tree. In each node of the tree one or more tokens can be placed. After the placement a node with two or more tokens can be selected and two tokens can be removed from the selected node and one token can be placed to any adjacent node to the selected one. Such operation can be repeated several times. Your task is to find the maximal number of tokens (modulo **M**) that can be placed on the tree nodes to fulfill the following condition: there exists at least one node, to which it is impossible to move a token applying given operations.

Input:
First line of input contains the quantity of tests **T** (**1≤T≤20**).
First line of each test case contains two numbers: **N** (**2≤N≤30000**) – the number of nodes in the tree and **M** (**2≤M≤**2^{31}**–1**). Then **N–1** lines follows, each of which contains **2** adjacent node numbers (nodes are numbered from **1** to **N**) separated by space.

Output:
Output **T** lines of the form "**Case #A: B**", where **A** is the number of test (beginning from **1**), **B** is the desired number for this test case.

Input	Output
2 6 997 1 2 1 4 3 4 5 3 3 6 7 13 1 2 1 3 1 4 2 5 3 6 4 7	Case #1: 16 Case #2: 5

Developed by Deep Vision