Local Selection Contest
for
ACM Greater New York Regional Programming Contest 2015

Stony Brook University
(Sep 15, 2015)

Rules.

1. There are seven (7) problems to be completed in three hours. Problems are *not* ordered according to their difficulty. You're strongly encouraged to read all problems before starting coding.

2. Solutions in C/C++/Java are supported. The options are:

	Version	Compiling	Running
C	GCC 5.2	`gcc -O2 -lm -o {basename} {basename}.c`	`./{basename}`
C++	GCC 5.2	`g++ -O2 -lm -o {basename} {basename}.cpp`	`./{basename}`
Java	Oracle Java 8	`javac {basename}.java`	`java {basename}`

   In the beginning of statements of each problem, {`basename`} will be given.

   For Java users: A Java solution must be named {`basename`}`.java`, contain a single public class {`basedname`}, in which there must be a public method `public static void main(String[] args)`, which is expected to be run by the judge. You can have other non-public classes and arbitrary nest classes. Also do *not* create packages in your code. Failing to follow the instructions will prevent your solution from running.

3. Your solution will be judged on a quad-core Intel Core i7 machine running 64-bit UNIX. However, you should write your code in a portable manner *without* assuming that it will always run in a 32 bit environment.

4. All problems require you to read test data from the standard input and write results to the standard output.

5. Output must correspond exactly to the provided sample output format, including (mis)spelling and spacing. Multiple spaces will not be used in any of the judges output, except where explicitly stated.

6. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.

7. Contestants will submit their program through the PC2 software.

8. Contestants can use their books, papers, documentation, source codes of programs. But no soft copy will be allowed.

9. Judges' decisions are to be considered final.

We more frequently fail to face the right problem
than fail to solve the problem we face.

Problem A

Auction

basename: auction

Time Limit: 2 seconds

Problem Description

I own a auction website that sells finest handmade luxury items (e.g., handmade swiss watches). We preorder each item in batches[1], and as soon as we place the order we start an auction for that batch. Since the items are carefully handmade, it takes some time to make each piece, and so we receive the items of the same batch in smaller sub-batches over a period of time.

If we place an order for n pieces of an item, we accept exactly n bids[2] for that batch. Each person can place at most one bid, and each bidder eventually receives a piece. However, since the pieces arrive in small sub-batches, the bidders end up receiving the pieces at different times depending on their bid amount and bidding time. For example, as soon as we receive a sub-batch of p pieces, we remove the highest p bidders from the current set of bidders one after another and send a piece to each of them in the order they are removed. In case there are fewer than p bidders we remove all of them, and set aside the remaining pieces of the current sub-batch to add to the last sub-batch we will receive. We assume that no two bidders place their bids at the same time, and in case two bidders have the same bid amount the one with the earlier bidding time gets preference.

This problem asks you to answer queries related to the order in which the bidders recieve their pieces.

Input Format

The first line of the input contains a positive integer $T(\leq 15)$ giving the number of test cases (i.e., auctions) to follow.

Each test case starts with a line containing three integers n, m and q in that order separated by spaces. The first integer $n \in [1, 150000]$ specifies the number of bidders in this auction. The second integer $m \in [0, n]$ gives the number of sub-batches received before the final sub-batch that contains all remaining pieces. The last integer $q \in [1, 100]$ is the number of queries to follow.

The next n lines contains information on the n bids in the order they were received. Each line contains the name of a bidder followed by his/her bid amount. The name is a string of at most 200 characters without any blanks, and the bid amount is a positive integer not larger than 10^8. The name and the bid amount are separated by spaces.

The next m lines contains information on the sub-batches (except the final one) in the order they were received. Each of these lines contains two integers $t \in [1, n]$ and $p \in [1, n]$ meaning that the

[1]Bulk purchase allows us to negotiate much lower price per piece with the manufacturer.

[2]Each auction has an associated minimum bid value, that is, no bid can be smaller than the minimum allowed bid value. However, we ignore that in this problem to keep things simpler.

sub-batch was received right after the t-th bid (but before the next bid, if any), and contained p pieces.

The last line of the test case contains q integers $n_1, n_2, \ldots, n_q$, where each $n_i \in [1, n]$ represents a query asking the name of the n_i-th bidder to receive his/her piece.

Output Format

For each test case output a single line containing the names of the bidders in the order they were asked. Seperate every two consecutive names by a space.

Sample Input

```
1
7 3 4
Arnold 3
Bradley 3
Catherine 3
Donna 5
Emily 6
Fred 4
Greg 2
1 1
4 2
6 2
1 2 4 6
```

Sample Output

```
Arnold Donna Emily Catherine
```

Problem B

My Fears

basename: fears

Time Limit: 2 seconds

Problem Description

First let me tell you about my fears. I am agoraphobic – in particular, I am afraid of entering rooms that have fewer than two exits (i.e., doors). I am also quodnumerophobic – meaning I do not like even numbers. Now I hope you will understand why I ask you to solve the following problem.

There many interconnected rooms[3] in each building in SBU. I will tell you how the rooms are connected as well as the area of each room. I simply want to know the total area of all rooms that I can enter without any fear.

This is what you need to do. You first mark each room that is connected to fewer than two other rooms as inaccessible. Then consider the network of the remaining rooms, and find all connected components. If a connected component has an even number of rooms you know that I will be afraid to enter any part of that component, and so you can mark all rooms in that component as inaccessible. Now sum up the areas of all remaining rooms (i.e., rooms that are not marked as inaccessible), and let me know.

Input Format

The first line of the input contains a positive integer $T(\leq 30)$ giving the number of test cases to follow.

Each test case starts with a line containing two positive integers n and m in that order separated by spaces. The first integer $n(\leq 10000)$ specifies the number of rooms, and the second integer $m(\leq 100000)$ gives the number of doors connecting a pair of rooms. The rooms are uniquely numbered using integers in $[1, n]$.

Each of the next n lines gives the area[4] of one room. The i-th integer gives the area of room i.

Each of the next m lines contains a pair of integers $a \in [1, n]$ and $b \in [1, n]$ meaning that rooms a and b share a door.

Output Format

For each test case output a line containg an integer that gives the total area of all rooms that I can enter without any fear.

[3] corridors are also counted as rooms

[4] using a commonly used unit which we have omitted

Sample Input

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

Sample Output

```
10
```

Problem C

Self-describing Pyramid

basename: pyramid

Time Limit: 2 seconds

Problem Description

Solomon Golomb's *self-describing sequence* is a nondecreasing sequence of natural numbers (i.e., positive integers) with repeations. For integer $i \geq 1$, the i-th number in this sequence is denoted by $g(i)$, where $g(1) = 1$ and each integer $i \geq 2$ appears in exactly $g(i)$ consecutive locations in the sequence. Figure 1 shows the first few numbers in this sequence.

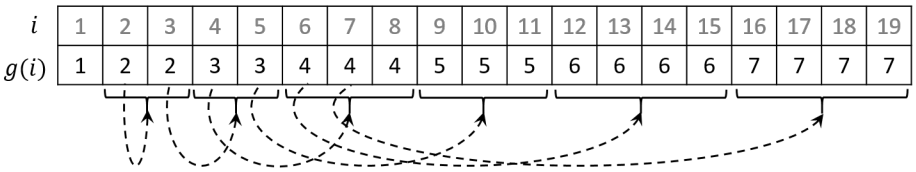

Figure 1: Solomon Golomb's self-describing sequence.

A *self-describing pyramid* of height $h \geq 1$, where h is an integer, consists of h layers numbered from 1 to h from top to bottom. The entire pyramid is made of $1 \times 1 \times 1$ blocks (unit cubes), and layer $i \in [1, h]$ contains exactly $i \times g(i)$ of them covering a $i \times g(i)$ rectangular area. Figure 2 shows a self-describing pyramid of height 7 containing exactly 90 unit cubes.

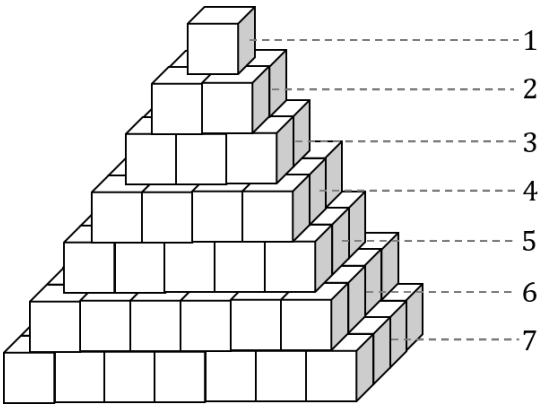

Figure 2: A self-describing pyramid of height 7 consists of exactly 90 unit cubes.

Given the height of a self-describing pyramid, this problem asks you to count the number of unit cubes it contains.

Input Format

The first line of the input contains a positive integer $T(\leq 2000)$ giving the number of test cases to follow.

Each of the next T lines contains an integer $h \in [1, 10^9]$ specifying the height of a self-describing pyramid.

Output Format

For each test case output a line containing an integer giving the number of unit cube the pyramid contains. If the number is greater than 1000000006 report the number modulo 1000000007.

Sample Input

```
4
4
10
6
100000
```

Sample Output

```
23
217
62
507231491
```

Problem D

A Piece of π

basename: pi

Time Limit: 2 seconds

Problem Description

The following is a mnemonic for the decimal expansion of π ($= 3.14159265358979323846264\ldots$). Each successive digit in π is the number of letters in the corresponding word below.

"$\underbrace{How}_{3}\ \underbrace{I}_{1}\ \underbrace{want}_{4}\ \underbrace{a}_{1}\ \underbrace{drink}_{5},\ \underbrace{alcoholic}_{9}\ \underbrace{of}_{2}\ \underbrace{course}_{6},\ \underbrace{after}_{5}\ \underbrace{the}_{3}\ \underbrace{heavy}_{5}\ \underbrace{lectures}_{8}$
$\underbrace{involving}_{9}\ \underbrace{quantum}_{7}\ \underbrace{mechanics}_{9}.\ \underbrace{All}_{3}\ \underbrace{of}_{2}\ \underbrace{thy}_{3}\ \underbrace{geometry}_{8},\ \underbrace{Herr}_{4}\ \underbrace{Planck}_{6},\ \underbrace{is}_{2}\ \underbrace{fairly}_{6}\ \underbrace{hard}_{4}\ldots$"

But π being an irrational number its expansion goes on and on and on, never repeating itself! All we can do is to approximate its value.

Here is a very simple way of approximating π. Given a positive integer r, first draw a circle of radius r centered at the origin of axes. Then count the number of grid points (i.e., points (x, y), where both x and y are integers) that lie strictly inside the circle or on its periphery. Let this number be n. Then

$$\pi \approx \frac{n}{r^2}.$$

For example, in Figure 3 below, $r = 10$, $n = 317$, and so $\pi \approx \frac{317}{10^2} = 3.17$.

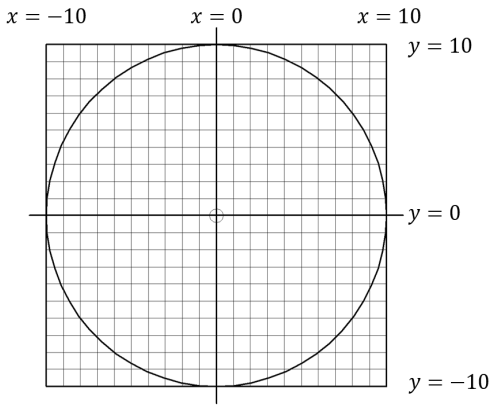

Figure 3: Approximating π by counting the number of grid points on or inside the circle.

Input Format

The first line of the input contains a positive integer $T(\leq 20)$ giving the number of test cases to follow.

Each of the next T lines contains an integer $r \in [1, 10000]$ specifying the radius of the circle to use for approximation.

Output Format

For each test case output a line containing the approximate value of π rounded to 10 decimal places calculated using the procedure described above.

Sample Input

```
3
10
150
5000
```

Sample Output

```
3.1700000000
3.1413777778
3.1415870800
```

Problem E

Desert Cities

basename: cities

Time Limit: 2 seconds

Problem Description

I am planning to tour a network of desert cities connected by (bidirectional) highways. I have a car, but there is not a single gas station along those highways. So refueling won't be possible until I reach a city. Depending on how much gas my car can carry (full tank + extra cans), some of the highways may simply be too long for it to travel. I say that (a, b) is a connected pair of cities for me provided I am able to travel from a to $b \neq a$ (possibly through other cities) without ever running out of gas on a highway. I will always fully refuel before leaving a city.

If I give you a description of the highway network connecting the cities, and tell you how many miles I can travel with the maximum amount of gas my car can carry, can you tell me how many pairs of cities are connected for me? Assume that (a, b) and (b, a) count as different pairs.

Input Format

The first line of the input contains a positive integer $T(\leq 5)$ giving the number of test cases to follow.

Each test case starts with a line containing three integers n (≤ 20000), m (≤ 100000) and q (≤ 5000) in that order separated by spaces. The first integer n specifies the number of cities, and the second integer m gives the number of highways, where each highway is bidirectional and connects a pair of cities. The cities are uniquely numbered using integers in $[1, n]$.

Each of the next m lines describes a highway using three integers $a \in [1, n]$, $b \in [1, n]$ and $d \leq 10000$ which means that the highway between cities a and b is d miles long.

Each of the next q lines contains an integer x asking for the number of connected city pairs under the constraint that my car cannot travel more than x miles with the maximum amount of gas it can carry.

Output Format

For each query print a line containing an integer giving the number of connected city pairs under the given constraint on my gas mileage.

Sample Input

```
1
6 7 3
1 3 275
```

```
1 5 160
2 2 200
2 3 70
3 5 45
4 3 120
5 6 180
50
110
150
```

Sample Output

```
2
6
12
```

Problem F

A Child's Play

basename: play

Time Limit: 2 seconds

Problem Description

You are given two stacks of disks. All disks have the same thikness, but not necessarily the same radius. Your task is to remove the minimum number of disks from the two stacks so that the two stacks look exactly the same[5]. You are not allowed to change the order of the disks that remain in the two stacks. Figure 4 shows an example.

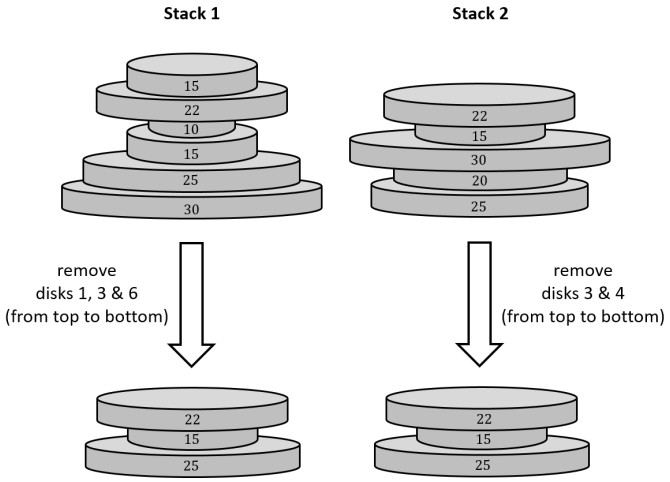

Figure 4: Removing the minimum number of disks from the two stacks to eliminate dissimilarities. Each disk is labeled with its radius.

Input Format

The first line of the input contains a positive integer $T(\leq 100)$ giving the number of test cases to follow.

Each test case starts with a line containing two integers n_1 and n_2 giving the number of disks in stacks 1 and 2, respectively. Assume that $1 \leq n_1, n_2, \leq 100$. The next line contains n_1 positive integers giving the radii of the disks (from top to bottom) in the first stack. Then follows another line containing n_2 integers giving the radii of the disks (from top to bottom) in the second stack. Each radius will be an integer in $[1, 1000]$.

[5]i.e., Both stacks will have the same number of disks with the same sequence of radii from top to bottom.

Output Format

For each test case output a line containing a nonnegative integer giving the number of disks remaining in each stack.

Sample Input

```
2
7 6
20 15 10 15 25 20 15
15 25 10 20 15 20
8 9
10 20 20 10 20 10 20 10
20 10 20 10 10 20 10 10 20
```

Sample Output

```
4
6
```

Problem G

Lucky Words

basename: lucky

Time Limit: 2 seconds

Problem Description

A word is considered lucky in our university if and only if it contains at least one copy of letter 's', 'b' and 'u', respectively. For example, 'sbu', 'uubs' and 'abxucps' are considered lucky, but 'abc', 'ssbb' and 'bs' are not.

Now you're given a set of words, and your task is to output the number of lucky words in this set.

Input Format

The first line of the input contains a positive integer $T(\leq 100)$ giving the number of test cases to follow.

Each test case is described by a single line containing a positive integer $n(\leq 100)$, the number of words, followed by n words. Each words will consist of only lowercase letters ('a', 'b', 'c' ... 'z'). The length of every word is positive and not larger than 100. No two words are the same. Every two consecutive tokens (the integer n and the following words) are separated by a single space.

Output Format

For each test case output a line containing a nonnegative integer giving the number of lucky words in this set.

Sample Input

```
3
1 sbu
2 abc uubbs
3 bbb sss uuu
```

Sample Output

```
1
1
0
```