IEEEXTREME PROGRAMMING COMPETITION 9.0

#IEEEXtreme

IEEEXTREME PROGRAMMING COMPETITION 9.0

24 October 2015

Think you can code?

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE's premier 24-hour coding battle for students around the world.

Prove it.

IEEE

IEEEXtreme Programming Competition 9.0

Follow IEEEXtreme on reddit

24 October 2015

Starting at 00:00:00 UTC

IEEEXtreme is a 24-hour online coding competition, within which a worldwide community of college and university students enjoy an engaging set of unique programming challenges.

WHO CAN COMPETE?

- Teams of up to three collegiate students who are current IEEE student members
- A local college or university may form multiple teams

WHERE IS THE COMPETITION HELD?

- IEEEXtreme is a virtual event, but teams often organize around their local IEEE Student Branch

WHAT COULD I WIN?

- Fame: Unlimited bragging rights and an item for your resume
- Fortune: The Grand Prize is a trip to the IEEE conference of your choice, anywhere in the world

Not an IEEE member?

If you are not a student member, you can join both IEEE and IEEE Computer Society for US$35 or US$40, depending on location. Current IEEE student members can add IEEE Computer Society Student Membership for just US$8.

www.computer.org/students

IEEE computer society

For more information, visit: **www.ieee.org/xtreme**

IEEE

Prizes

All active participants in the competition will receive a digital certificate and digital gift. "Active participant" is described as a team who makes a reasonable attempt at solving a problem.

1st Place:

The winning team members will receive an expenses-paid trip to an IEEE conference of their choice, anywhere around the world. Roundtrip coach airline tickets for each winner from winner's preferred major metropolitan airport to the conference destination, conference registration fees, and a three night hotel stay (confirmation pending) will be provided by IEEE for winning team members.

2nd Place:

Each member of the team that wins 2nd place in the IEEEXtreme 9.0 competition will receive an iPad Air.

3rd Place:

Each member of the team that wins 3rd place in the IEEEXtreme 9.0 competition will receive an iPad Mini.

4th - 10th Place:

Each member of the 4th through 10th place teams in the IEEEXtreme 9.0 competition will receive a Raspberry Pi computer.

Top 100:

All members of teams that place in the top 100 at the end of the competition will receive a special edition IEEEXtreme 9.0 gift bundle, including a reserved IEEEXtreme "Top Coder" t-shirt and a valuable "Essential Studio Enterprise Edition" license for a comprehensive development suite that includes 650+ components across multiple platforms. This gift is courtesy of our sponsorship partner Syncfusion.

Taxes, if any, are the sole responsibility of each winning team member.

Void where prohibited by Federal Law.

IEEEXTREME PROGRAMMING COMPETITION 9.0

The Organizers

Executive Leadership

Dimitrios Lyras, IEEEXtreme 9.0 Project Lead
Germany

Dr. Dimitrios Lyras was born in Kozani, Greece. He received his diploma in Electrical and Computer Engineering and his Ph.D. in Data Mining and Deductive Logic Reasoning from the University of Patras, Greece. He is presently working as a post-doctoral researcher at the Ludwig-Maximilians-University of Munich, Germany, in the field of multiple sequence alignment.

Sinan AlSheikh, IEEEXtreme 9.0 Ex-Lead and Project Adviser
United Arab Emirates

Sinan AlSheikh is currently working as an IT consultant with IBM. As an undergraduate student, he filed his first US patent. AlSheikh earned his M.S. in Informatics from the British University in Dubai, and his B.S. in Electrical Engineering from the University of Sharjah.

Jeremy Blum, IEEEXtreme 9.0 Technical, Industry, and Judges Lead
US

Dr. Jeremy Blum is an associate professor of Computer Science at Pennsylvania State University, Harrisburg, PA, US. This will be the fifth Xtreme competition for Dr. Blum. He served as a proctor for Xtreme 5.0 through 7.0, and as a judge for Xtreme8.0. In addition to his participation in IEEEXtreme.

Prasanth Mohan, IEEEXtreme 9.0 Public Relations Co-Lead
India

Prasanth is currently working as an associate engineer at SP Robotic Works. He obtained his bachelor's degree from Sri Muthukumaran Institute of Technology affiliated to Anna University, Chennai. He has been an active member of IEEE for the past three years. Prasanth obtained the Richard E. Merwin Scholarship from the IEEE Computer Society for the year 2013.

Dinko Jakovljevic, IEEEXtreme 9.0 Public Relations Co-Lead
Croatia

Dinko Jakovljevic is a graduate student at University Of Josip Juraj Strossmayer in Osijek where he studies Computer Engineering. Dinko is Award & Contest Coordinator in Region 8 SAC. Two years ago, he co-founded IEEEmadC (Mobile Application Development Contest). Dinko received Richard E. Merwin Scholarship from IEEE Computer Society.

IEEEXTREME PROGRAMMING COMPETITION 9.0

Returning judges

Dr. Marco L. Della Vedova
Italy

Dr. Marco L. Della Vedova (S'10, M'14) is assistant professor of computer science at the University of the Sacred Heart in Brescia, Italy. He studied computer engineering at the University of Pavia, Italy, where he graduated in 2009 and received the Ph.D. degree in 2013. He held a visiting position at the University of California, Berkeley, USA, in 2011. He is a (co)author of 10+ scientific articles, published in peer-reviewed conferences and journals. He organized the first-ever IEEEXtreme event at the University of Pavia in 2011 and participated in all the following editions, first as student, then as proctor, and then as QA.

Dr. Vincent Gripon
France

Vincent obtained his M.S. from École Normale Supérieure of Cachan and his Ph.D. from Télécom Bretagne. His research interests include information theory, neuroscience, and theoretical and applied computer science. His intent is to propose models of neural networks inspired by information-theory principles, what could be called informational neurosciences. He is also the co-creator and organizer of an online programming contest named TaupIC, which targets French top undergraduate students. He ranked fifth out of 1515 in IEEE Xtreme programming competition in 2011.

Dr. Ashad Kabir
Australia

Dr. Kabir is a Teaching Scholar in the School of Information Technology at Deakin University, Melbourne, Australia. He is also an Adjunct Research Fellow in the Centre for Computing and Engineering Software Systems at Swinburne University of Technology, Melbourne, Australia. Dr. Kabir has research experience in a range of areas of computer science. He has participated and ranked top position in a number of national and international programming contests including the IEEExtreme Programming Contest and ACM ICPC Programming Contest.

Alberto Lorente Leal
Sweden

Alberto Lorente is a Young Professional IEEE member currently working as a software developer at Comeon! in Stockholm, Sweden. As an undergraduate student, he published his first paper in the IEEE EDUCON congress in 2010. Alberto earned both his M.S. in Software Engineering of Distributed Systems from the Royal Institute of Technology, KTH in Sweden and M.S. in Telecommunications Engineering from the Technical University of Madrid, UPM, in Spain in 2013. Alberto is an active IEEE member in the Sweden section, as chair of Young Professionals Affinity Group.

Dr. Oded Margalit
Isreal

Dr. Oded Margalit earned his Ph.D. in Computer Science from Tel Aviv University in 1993 under the supervision of Professor Zvi Galil. He has been serving as a judge since 2008, and he was awarded the IEEE MGA Achievement Award in 2009. Currently he is the CTO of IBM's Cybersecurity Center of Excellence in Beer Sheva, Israel and is the author of IBM's Ponder-This monthly challenge site. An avid riddle and puzzle maven, Margalit has been known to focus this passion on setting up contests and challenges for coding, puzzle-solving, and mathematical challenges.

IEEEXTREME PROGRAMMING COMPETITION 9.0

Returning judges

Anoop Thomas Mathew

India

Anoop Thomas Mathew, a.k.a. ATM, has been a judge of IEEEXtreme competition for the past four versions. As his day job, he heads the technology at data-crunching startup Profoundis and builds Vibe app. He has spoken at conferences like Fifth Elephant 2012, FOSSMeet 2011, PyCon 2012, FOSSMeet 2013, and PyCon 2013, to name a few. He loves open source and codes mostly in python and JavaScript. Connect with him on Twitter @atmb4u.

George Michael (Γιώργος Μιχαήλ)

Cyprus

George Michael is an enthusiastic software developer at XM.COM. He has a B.Sc. and an M.Sc. in Computer Science from the University of Cyprus. He worked at the Data-Driven Multithreading Laboratory at the University of Cyprus for a few years doing research on high-performance computing. His work is mostly on GNU/Linux. He is an active IEEE volunteer and he held various positions during the past few years. Between them: member of the board of the University of Cyprus IEEE Student Branch/IEEE Cyprus Section Executive Committee/IEEE Region 8 Committee.

Charalampos Tsimpouris

Greece

Charalampos Tsimpouris was born in 1985 and graduated in June of 2008 from the Department of Electrical & Computer Engineering at University of Patras. Since October 2008, he has been working as a Ph.D. student at the same department, under Professor's Sgarbas expertise, working in the area of AI and computational linguistics on Greek law texts. He has participated in the following programming contests: 10th Balkan Olympiad in Informatics, held in Belgrade; and 2002, 14th International Olympiad in Informatics, held in Yong-In, South Korea. He has also participated successfully as a judge for IEEEXtreme 6.0, 7.0 and 8.0.

Dario Schor

Canada

Dario received his M.Sc. in Computer Engineering under the supervision of Prof. Witold Kinsner from the University of Manitoba, Winnipeg, Canada, in 2013. That summer, he also completed the Space Studies Program from the International Space University in Strasbourg, France. He is currently working as a software engineer in the Space Division of Magellan Aerospace, Winnipeg, developing flight software and ground tools for the Radarsat Constellation Mission. He has been an IEEE member since 2007. He participated in IEEEXtremes 3.0, 4.0, and 5.0, and he most recently served as a mentor for Xtreme 7.0.

Kbs Naveen

India

"Kundula Bala Satya Naveen (KbsNaveen) is currently working at Amazon India Pvt Ltd, Hyderabad, India. Prior to Amazon he was working for Infor India Pvt Ltd (Hyderabad) and before that he worked for Broadcom (Bangalore), which was his first full time job. He graduated as Computer Science Engineer in the year 2013. He is a keen programmer. He has been participating in online programming contests/competitions for the last 2 years. Apart from that Naveen also enjoy reading science and technical articles, plays badminton, chess and cricket.

New judges

Dr. Linda Null

U.S.A.

Dr. Linda Null is an associate professor of Computer Science at The Pennsylvania State University, Harrisburg, where she has taught since 1995. In addition, she serves as Associate Program Chair of Mathematics and Computer Science and Graduate Coordinator for Computer Science. She received both a Ph.D. and an M.S. in Computer Science from Iowa State University, as well as an M.S. in Computer Science Education, an M.S. in Mathematics Education, and a B.S. in Mathematics and English, all from Northwest Missouri State University. Dr. Null was recently presented Penn State's Award for Excellence in Teaching in recognition of her innovative and outstanding work in the classroom, as well as the Kathryn Towns Award in recognition of her commitment to the issues and interests of women students, particularly those in mathematics and computer science. She has also won three Texty awards for her computer organization and architecture textbook.

Dr. Tim Wahls

U.S.A.

Dr. Tim Wahls is an Associate Professor of Computer Science at Dickinson College in Carlisle, Pennsylvania. His research interests include programming languages, formal semantics, formal software engineering methods and generating code from formal models. His teaching interests additionally include algorithms, constraint programming, databases and animal rights. He received his Ph. D. in Computer Science from Iowa State University, under the supervision of Drs. Gary Leavens and Albert Baker.

Dr. Carol Wellington

U.S.A.

Carol Wellington has earned B.S., M.S., and PhD degrees in Computer Science from University of Delaware, Villanova University, and NC State University. She currently serves as a Software Engineering Professor at Shippensburg University of Pennsylvania and as Chair of their Computer Science & Engineering Department. Her industry experience includes operating system development and real-time telecommunications software development. Her PhD research was in learning and reasoning under uncertainty in artificial intelligence.

IEEEXTREME PROGRAMMING COMPETITION 9.0

New judges

Bjarki Ágúst Guðmundsson

Iceland

Bjarki is starting his MSc studies at Reykjavik University, Iceland, where he recently finished his BSc in Computer Science and Discrete Mathematics. He is very passionate about competitive programming, and spends most of his spare time practicing and competing. Bjarki competed in IEEEXtreme 6.0, 7.0 and 8.0 with different teams, and placed 21st, 4th and 1st, respectively.

Dr. Barry Wittman

U.S.A.

Dr. Barry Wittman is an associate professor of Computer Science at Elizabethtown College. He earned his Ph.D and M.S. in Computer Science from Purdue University and his B.S. in Computer Science from Morehouse College. His doctoral research was on approximation algorithms for routing problems, but he has since focused primarily on computer science pedagogy and the development of the Shadow programming language. Dr. Wittman has coached programming teams for the ACM ICPC as well as judged and written questions for regional programming competitions.

Tómas Ken Magnússon

Iceland

Tomas recently graduated with a BSc in Computer Science and Discrete Mathematics from Reykjavik University, Iceland and will start his MSc studies at the same institute in the coming fall. Competitive programming has been a big drive for his studies and sparked interest many fields of mathematics and computer science. Tomas has competed in the ACM-ICPC and last two years in IEEEXtreme, placing 1st last year.

IEEEXTREME PROGRAMMING COMPETITION 9.0

Quality Assurance

Siddharth Dahiya

U.S.A.

Siddharth Dahiya is a Software Engineer in Microsoft Office in Redmond, Washington. His research interests include scheduling and optimization problems. He received his Master's in Computer Science from The Pennsylvania State University, with his Master's Thesis titled "Course Scheduling with Preference Optimization" under Dr. Thang N. Bui.

Luis Fernandes

Canada

Dr. Luis A. Fernandes received a degree in Applied Physics in 2006 and a Ph.D. in Physics in 2012 from the Faculty of Science at the University of Porto, Portugal. In 2008 he was awarded a Ph.D. fellowship from the Portuguese government to pursue his thesis entitled "Birefringence and Bragg grating control in femtosecond laser written optical circuits". As a graduate student he was awarded a SPIE Scholarship in Optical Science and Engineering and an award for best oral presentations in 2010. At the end of 2013 he joined OZ Optics Ltd. as a senior optical scientist. He has authored and co-authored many scientific articles on optics and photonics. As a student, he participated in the IEEEXtreme competition in 2009, 2010, and 2012. He served as a member of the board of the University of Toronto SPIE Student Chapter.

Christopher Jackson

U.S.A.

"Christopher Jackson received his M.Sc. in Computer Science from the Pennsylvania State University, Harrisburg, PA, USA, in 2013. Christopher is currently working in the IBM Analytics group at IBM. He works on Big Match for Hadoop, an analytics tool used to derive insight on structured and unstructured data.

As a student Christopher personally competed in the IEEEXtreme 5.0 competion as well as many other local and online programming competitions. "

Quality Assurance

Pattanapoom Phinjirapong

U.S.A.

Pattanapoom received his bachelor degree in Computer Engineering at Chulalongkorn University, Thailand. He attended Penn State Harrisburg, PA, USA for his master degree in Computer Science during 2013-2015. He is going to work at Amazon.com.

Sachin Bharadwaj S

India

"Sachin Bharadwaj S is working as a Software Engineer at Analog Devices, Inc. in Bangalore, India. He has graduated with a bachelor's degree in Electronics and Communication Engineering from BMS College of Engineering, Bangalore in 2014. He has previously interned at Aeronautical Development Establishment and Raman Research Institute. He has also made contributions to the open-source projects like Wiselib under Google Summer of Code.

Sachin has been involved with IEEE as a student volunteer since 2011, when he was an undergraduate student. He was involved in activities related to IEEEXtreme in his IEEE student branch, and also been part of a competing team in IEEEXtreme for editions 5.0, 6.0, and 7.0.

Sachin is very enthusiastic about programming. He has participated in a number of national and international programming contests over the past 5 years including the ACM International Collegiate Programming Contest and has won several awards."

Martin Tribo

U.S.A.

"Martin Tribo is a software developer at Interactive Network Technologies (INT), Inc. in Houston, TX. He graduated from Christian Brothers University in Memphis, TN with a BS in Electrical Engineering and a BS in Computer Science.

Martin has been a member of IEEE since 2009. As a student member, he managed his branch's website and helped organize events such as SPAC. For IEEEXtreme, he garnered teams and interest at his university, competing in the 4.0, 5.0 and 6.0 competitions. As a professional member he proctored for the 7.0 and 8.0 competitions.

Passionate about programming, Martin has experience in multiple languages including C, C++, Java, Python and JavaScript. He currently develops HTML5 visualization tools for the oil and gas industry at his job. He also has experience working with systems such as: relational and nosql databases; messaging services; enterprise integration platforms. "

Info

Follow IEEEXtreme on

IEEEXtreme 9.0
Media and Press Kit

www.ieee.org/Xtreme | IEEEXtreme@ieee.org

Kit Contents:

- Introduction to IEEEXtreme
- IEEEXtreme 9.0 Promotional Flyer
- IEEEXtreme Media and Web Assets
- Sample Press Release for Local Advertising
- IEEEXtreme Facts and Figures

Introduction

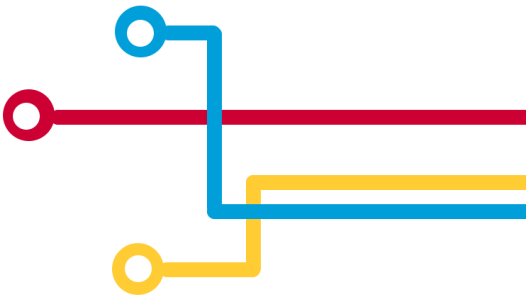
IEEEXtreme is a global challenge in which teams of IEEE Student members, supported by an IEEE Student Branch, advised and proctored by an IEEE member, compete in a 24-hour time span against each other to solve a set of programming problems.

IEEEXtreme 9.0 will be held 24 October 2015 00:00:00 UTC and teams' registration opens 17 August, 2015.

Included in this press kit is a promotional flyer that can be printed and displayed at local colleges and universities, a list of useful media and web assets, sample language for local advertising, and IEEEXtreme facts and figures.

Any questions concerning IEEEXtreme can be emailed: IEEEXtreme@ieee.org

Frequently asked questions and answers page: http://goo.gl/a24B9a



Follow IEEEXtreme on

IEEEXtreme Media and Web Resources

Media Resources

- IEEEXtreme Prezi

http://prezi.com/kstsylczt4cx/?utm_campaign=share&utm_medium=copy&rc=exoshare

- *The Institute* Magazine—"IEEEXtreme Winners: Where Are They Now?"

http://theinstitute.ieee.org/people/students/ieeextreme-winners-where-are-they-now

- Students Explain IEEEXtreme

https://www.youtube.com/watch?v=1OJf3OV-3BQ

Web Resources

- Official Website

http://www.ieee.org/xtreme

- Collabratec

https://goo.gl/fZ04HE

Facebook—Photos of Past Events Available

https://www.facebook.com/IEEEXtreme

- Twitter

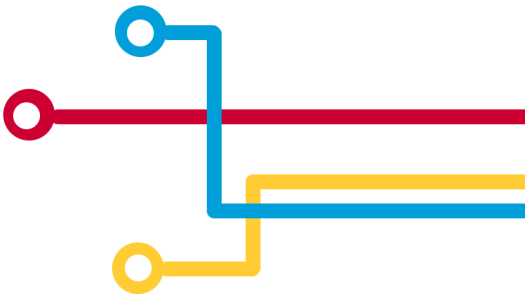
https://twitter.com/ieeextreme

- LinkedIn

https://goo.gl/EPKmlQ

- Wikipedia

https://twitter.com/ieeextreme



Follow IEEEXtreme on

Sample Press Release for Local Advertising

Local Students to Compete in IEEEXtreme Worldwide Programming Competition

Local undergraduate and graduate computer programming students are gearing up to compete for the rights to call themselves the "world's best programmers"—and a free trip to one of hundreds of IEEE's global technical conferences—during the upcoming programming competition, IEEEXtreme 9.0.

The ninth annual worldwide college programming competition involves teams of IEEE student members working collaboratively to solve a series of challenging programming problems. IEEE, the world's leading technical professional association for the advancement of technology, conducts the computer programming competition. Local colleges and IEEE student branches will host the competition on their local campuses.

The 24-hour competition will begin simultaneously around the world on 24 October, 2015 at 00:00:00 UTC . The registration deadline is 12 October, 2015. More information about IEEEXtreme 9.0, including an online registration form, is available at: http://www.ieee.org/xtreme.

About IEEE

Through its more than 400,000 members in 190 countries, IEEE is a leading authority on a wide variety of areas ranging from aerospace systems, computers and telecommunications to biomedical engineering, electric power and consumer electronics. Dedicated to the advancement of technology, IEEE publishes 30 percent of the world's literature in the electrical and electronics engineering and computer science fields, and has developed over 900 active industry standards. The organization annually sponsors more than 850 conferences worldwide. Additional information about IEEE can be found at http://www.ieee.org.

IEEEXTREME PROGRAMMING COMPETITION 9.0

Follow IEEEXtreme on

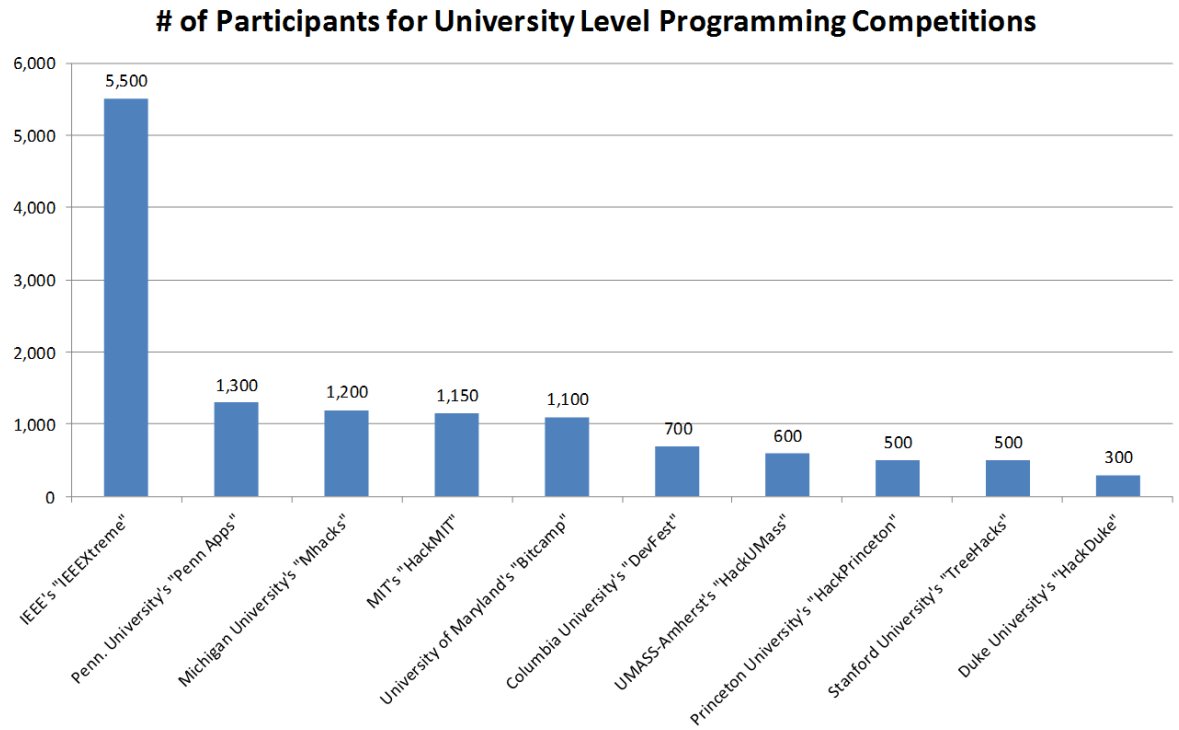
IEEEXtreme Facts and Figures

- Last year, IEEEXtreme 8.0 saw over 5,500 active participants in the global coding challenge; India, USA, Sri Lanka, Tunisia, Canada, and Greece were most active.

IEEEXtreme 8.0 Participating Countries

Rank	Country	Number of Teams Participating
1	India	456
2	United States	232
3	Sri Lanka	199
4	Tunisia	82
5	Canada	69
6	Greece	64

IEEEXtreme 8.0 Compared to Recent Elite Hackathons

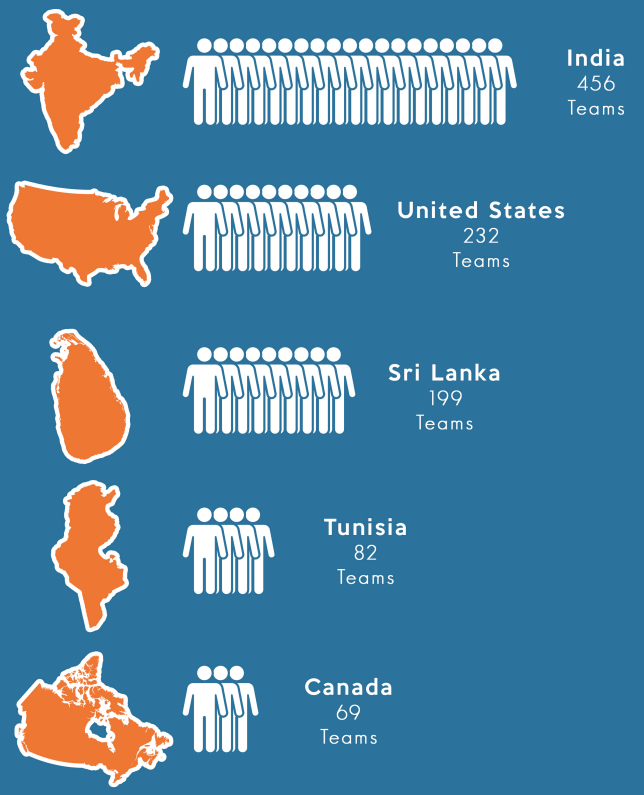


IEEEXtreme
in Numbers

2006 Xtreme started in 2006
- Number of teams: 44

2014 Xtreme 8.0
- Teams from 62 countries, from every IEEE region
- 1,853 actively competing teams

Top Participating Teams from Xtreme 8.0



India
456
Teams

United States
232
Teams

Sri Lanka
199
Teams

Tunisia
82
Teams

Canada
69
Teams

Top Performing Universities from Xtreme 8.0

The Rules

Date: 24 October 2015

Time: 00:00:00 UTC – 23:59:59 UTC

More Info: www.ieee.org/xtreme

IEEEXtreme 9.0 Competition Rules

Description

IEEEXtreme is a global challenge in which teams of student members, supported by an IEEE Student Branch, advised and proctored by an IEEE Member, compete in a 24-hour time span against each other to solve a set of programming problems.

The competition was created to:

- Provide IEEE Student Members with a new and interesting activity
- Give IEEE Student Members a challenge to embrace team work - an important skill to develop for career success
- Increase the number of IEEE Student activities with a focus on the computer, programming and information technology fields

Other benefits include providing Student Branches with ways to get IEEE Student Members involved in local activity in a fun and engaging way.

Sponsor

The IEEEXtreme Programming Competition is hosted by IEEE, 445 Hoes Lane, Piscataway, New Jersey, USA, 08854. It is organized and managed by the Student Activities Committee under the Member and Geographic Activities business unit of IEEE.

Eligibility

Participants must compete as part of a team. Teams are comprised of **up to** 3 IEEE student or graduate student members, but should only include a maximum of 2 graduate student members per team. All team members must be IEEE student or graduate student members to register and compete in the competition. IEEE Membership numbers are required during the registration process.

Universities and Colleges can have multiple teams.

Each team must have a proctor who will supervise during the 24-hour programming competition.

Team members must solve and complete the problems without assistance from others. Please note that the intent and spirit of the competition is for the students, not others, to solve a problem. Persons acting as proctor must limit the level of support and must not contribute in any other form that might be considered original authorship, or in any way that may enable claims of rights or ownership to the submitted entries. In no case will work-on-behalf of teams or individuals be allowed.

Void where prohibited by Federal Law.

Registration

Registration will be open between 17 August 2015 and 12 October 2015 (00:00:00 UTC).
Teams can find registration information at http://www.ieee.org/xtreme

Proctors

Each team must have a proctor to supervise competition activities.

Proctors must be an IEEE Member of higher membership grade. Student or Graduate Student Members are not allowed to proctor IEEEXtreme but are encouraged to participate as a team member in the competition.

Proctor information (IEEE Member Number) is required during the registration process.

Student Branch Counselors or Department Chairs make great Proctors as they are all higher grade IEEE members. Many IEEE Young Professionals are also higher grade IEEE members and may be eligible to serve as proctors.

Teams may want to recruit two or more proctors so that one can take a break to rest during the 24 hour competition.

Proctor tasks include:

- Monitor the general flow of the activity
- Inform students when the competition begins, at the middle of it, when there are 6 hours left and when there is 1 hour left
- Ensure that no one external to the team members helps or assists the student participants in resolving the problems in any way
- Responsible for the receipt and distribution of prizes for student teams

If you need assistance in finding a proctor, please consult our Guide on Finding a Proctor (PDF).

Please note: A Proctor can support up to 8 student teams and all of the competition participants under an individual proctor's supervision must compete in the same venue. For more information on venue, see the following section.

Venue

As IEEEXtreme is a virtual online competition, a physical location, or venue, must be identified for participants to use during the 24-hour competition.

Venues can be in an IEEE Student Branch office or a college lab or another location on campus. It must be a place that participants can use for the entire 24 hours during the competition and should be equipped with at least one computer and some type of connection to the internet.
A proctor must be physically located within the venue at all times throughout the 24-hour competition.

Student Branch Activity

Student Branches, if able, should support and help to the participating teams, helping locate an appropriate venue for use during the competition, promoting the competition, assisting in identifying appropriate proctors, and increasing awareness of the student branch presence on campus.

Students attending universities who do not have an IEEE student branch on campus can still participate in the IEEEXtreme competition. This is an opportunity to bring students together to have fun with IEEE activities. Consider using this opportunity as a way to form a student branch. More information on how to form a student branch can be found here.

Problems

Problems are developed and judged by expert programmers. The panel of judges is made up of higher grade IEEE members from both Academia and Industry backgrounds.
Problems will be categorized as easy, moderate, difficult, advanced, and Xtreme to allow for students of all experience levels to participate.

All of the problems can be answered in any of the supported languages, which are indicated in the table below. The time and memory limits will apply to problems, unless otherwise specified in the problem definition.

Language	Version	Standard Challenges		
		Time limit in seconds	Memory limit in MB	Libraries provided
C	gcc 4.9.2, C99 standard	2	512	Math library -lm json library
C++	g++ 4.9.2, C++11 standard	2	512	Math library -lm json library
C#	Mono C# compiler 3.2.8.0.NET 4.0 CLR	3	512	newtonsoft json library
Python	Python 2.7.6	10	512	
Python 3	Python 3.4.0	10	512	
Java	Sun Java 1.7.0_55	4	512	Name your class Solution json-simple json library
Java 8	Sun Java 1.8.0_05	4	512	Name your class Solution json-simple json library
PHP	PHP 5.5.9	9	512	

Perl	Perl (v.5.18.2)	9	512	json library
Ruby	Ruby 2.0	10	512	
Objective-C	Objective-C 2.0: clang 3.4-1	2	512	Runtime(gnustep-libobjc2) Foundation Kit Blocks runtime libdispatch
Haskell	Ghc 7.8.4	5	512	logict lens pipes mwc-random hashtables regex-pcre hmatrix aeson and hashmap libraries are available.
Clojure	Clojure 1.6.0	8	512	
Scala	Scale 2.11.0	7	512	Have your entry point inside an objected name Solution
Common Lisp (SBCL)	SBCL 1.2.3	12	512	
Lua	Lua 5.2.3	12	512	
Erlang	Version 6.3	12	512	Have your main function in module solution
Javascript	Node v0.10.28	10	512	
Go	Go1.4	4	1024	
Groovy	1.8.6	5	512	JVM: 1.7.0_55
OCaml	Ocamlopt, version 4.01.0	3	512	Jane Street OCaml core libraries
F#	Fsharp 3.0.34, Mono 3.2.8	4	512	
VB.NET	Mono 3.2.8.NET 4.0 CLR	5	512	
LOLCODE	Version 1.2 with lci v0.10.5	5	512	
Smalltalk	GNU Smalltalk 3.2.4	5	512	
Tcl	Version 8.5 with tclsh	5	512	
R	Version 3.0.2	3	512	
RACKET	Version 6.1	10	512	
RUST	Version 1.0	5	512	
SWIFT	Version 1.2	2	512	Foundation
PASCAL	Version 2.6.2-8	2	512	
BASH	Version 4.3.11	1	512	
D	Version 2.067.0-b1	3	512	

Sample problems from previous competitions can be found at: http://www.ieee.org/membership_services/membership/students/awards/xtremesamples.html.

A demo practice contest community can be found at: https://www.hackerrank.com/contests/ieeextreme-challenges/challenges.

Problem Submission

Teams should submit their problem solutions electronically using the contest management tool. Instructions on access and utilization of the contest management tool will be provided to teams after registration closes.

The 2015 contest problems will be available the day of the contest.

Scoring Criteria

Simply put, if you solve a problem correctly, you get 20 points. You can gain 80 extra points depending on how difficult the problem is. The difficulty of any problem comes from how many other teams solved the same problem. If a lot of other teams solved the same problem that means the problem is easy and you will not get extra points on it. However, if you and few other teams solved a problem that means the problem is very hard and your team deserves more points on it.

This way, we advise you not to share your solution with other teams, because it will harm your score.

Note: Time is not directly included in the scoring formula. While it is used to break ties, you can take your time and solve the problem correctly. Moreover, you should also note that the number of unsuccessful attempts to solve a problem will not harm you score but it will indicate that the problem is hard and that will help improve other teams who solved the problem successfully. So try to be one of the smartest teams who solves the problem first and let all other teams improve your score ☺

$$\textbf{Problem Score} = 20 + max\left(0,80\ \left(1 - 2\ \frac{Successful\ Attempts}{Total\ Attempts}\right)\right)$$

Partial scoring:

Sometimes, you write the perfect code that passes all test cases except the last one and you don't know why. Let's say you attempted to solve problem X, which has cases 1 to 5, and successfully solved cases 1, 3 and 4. The score you get will be a weighted factor of the three cases you were able to solve. If you cracked all 5 the fraction will be simply '1', in which case you decrease the score of all other teams who were able to solve the same problem. Otherwise, you will be just increasing them.

Each test case is assigned a weight. The sample test cases have a very small weight, while the hidden test cases have larger weights. Thus, submitting a solution that solves only the sample test cases will earn only a very small score.

$$Submission\ Score = problem\ score * \frac{Sum\ of\ correct\ test\ cases'\ weights}{Sum\ of all\ test\ cases'\ weights}$$

Rank is decided upon score. However, terms of draw time will be considered as a factor to rank teams. Ex: Team A and Team B can have same score, let's say X, but then have different ranks, say Rank 2 and Rank 3. This means the Team with higher rank was faster to solve problems compared to the other team.

Tie Breakers:

In the event that two teams are tied, the tie will be broken based on which team has the smaller total submission time. This time is equal to sum of the elapsed time for the best submission to each problem, ignoring the problem score. For example, let's say that a team makes submissions as shown in the table below.

Submission Number	Problem Number	Submission Time	$\frac{\text{Sum of correct test cases' weights}}{\text{Sum of all test cases' weights}}$
1	1	1 am, UTC	0.20
2	2	2 am, UTC	0.90
3	1	3 am, UTC	0.70
4	1	4 am, UTC	0.70
5	1	5 am, UTC	0.60

To calculate the total submission time for the example, we consider the best submissions for each problem. For problem 2, the best (and only) submission occurred 2 hours into the contest. For problem 1, the best submission occurred 3 hours into the contest. Note that if an identically scoring submission occurred later, we use the earliest of these identical submissions. Therefore, we ignore submission number 4 and 5 because neither of these were an improvement over submission number 3. In this case, then, the total submission time for the team would be 5 hours.

Reminders:

No language has any advantage over the others. (Ex: Java, C, Python, PHP, etc. are all the same). Only the problem submission will impact the score, compiling will not affect your score at all.

Your score can be different when you wake up. So, don't lose your hope and don't be so confident ☺. Most importantly, ENJOY IEEEXtreme!

Supported Browsers

The browsers that are supported to run IEEEXtreme 8.0 are as follows:

- Chrome v 44
- Firefox v 39
- IE 11

Please consult each browser's Web site for more information on updates.

Selection of Winners

Winners are determined strictly based on overall score as determined by the scoring outlined above. As noted above, in the case of a tie, time will be considered as a factor to rank teams.

Notification of Winners and Final Rankings

From the close of the competition through 31 October the IEEEXtreme Technical team will be evaluating code submissions. IEEE reserves the right to disqualify a team if it's found to have manipulated or cheated during the competition. The official results will be communicated on or about 2 November. Winners will be contacted by IEEE directly.

Requirements of Winners

IEEE may, within its sole discretion, require each prize winner to sign and return an affidavit of eligibility and liability and publicity release, in which each winner consents to the use of his or her name, age, hometown and photo by IEEE for advertising and promotional purposes, without any additional compensation, wherever lawful, as a precondition to award of a prize. If any prize winner fails to sign and return the requested affidavit of eligibility and liability/public release as requested by IEEE, that winner may be disqualified, and his or her prize will thereafter be awarded to an alternate winner from the remaining valid entries using the criteria specified above. IEEE may also require each winner to assign all rights in any submission that is chosen as a winner to IEEE as a precondition to award of a prize. If any prize winner fails to assign all rights in the selected submission to IEEE, the winner may be disqualified, and his or her prize will thereafter be awarded to an alternate winner from the remaining valid entries using the criteria specified above. All prizes, including the travel arrangements for first place winners, must be claimed within one calendar year of the competition.

Prizes

All active participants in the competition will receive a digital certificate and digital gift. "Active participant" is described as a team who makes a reasonable attempt at solving a problem.

Prizes for IEEEXtreme 9.0

1st place: The winning team members will receive an expenses-paid trip to an IEEE conference of their choice, anywhere around the world. Roundtrip coach airline tickets for each winner from winner's preferred major metropolitan airport to the conference destination, conference registration fees, and a three night hotel stay (confirmation pending) will be provided by IEEE for winning team members.

2nd place: Each member of the team that wins 2nd place in the IEEEXtreme 9.0 competition will receive an iPad Air.

3rd place: Each member of the team that wins 3rd place in the IEEEXtreme 9.0 competition will receive an iPad Mini.

4^{th}-10^{th} place: Each member of the 4^{th} through 10^{th} place teams in the IEEEXtreme 9.0 competition will receive a Raspberry Pi computer.

"Top 100: All members of teams that place in the top 100 at the end of the competition will receive a special edition IEEEXtreme 9.0 gift bundle, including a reserved IEEEXtreme "Top Coder" t-shirt and a valuable "Essential Studio Enterprise Edition" license for a comprehensive development suite

that includes 650+ components across multiple platforms. This gift is courtesy of our sponsorship partner Syncfusion."

Taxes, if any, are the sole responsibility of each winning team member.

Void where prohibited by Federal Law.

Use of Entries

No entries will be returned. All entries become the property of IEEE. By entering, all participants consent to the use by IEEE of all the information provided in the entries for marketing or sales promotion purposes without any attribution, identification, right of review or compensation. All entrants agree to release and hold harmless IEEE and its officers, directors, employees and agents from and against any claim or cause of action arising out of participation in the contest.

Disputes Concerning the Competition

These rules shall be construed and governed by the laws of the State of New Jersey. Participants hereby consent to the personal jurisdiction in and venue of the courts located in the State of New Jersey for the adjudication of any and all claims arising out of or relating to the subject matter of this contest, and the interpretation or enforcement of the official rules thereof.

Funding Sources

The IEEEXtreme 9.0 Competition is being underwritten by IEEE Membership and Geographic Activities Department.

Corporate Sponsorship opportunities are still available. Please contact ieeextreme@ieee.org for more information.

Agreement to the Official Rules

By participating in this contest, participants agree to abide by the terms and conditions as established by IEEE. IEEE reserves the right to qualify all submissions and to reject any submissions that do not meet the requirements for participation as established by IEEE.

Contest Results and Official Rules

To obtain the names of any winners and/or a copy of these Official Rules, send a self-addressed, stamped envelope to IEEEXtreme 9.0 Competition, Member and Geographic Activities, IEEE, 445 Hoes Lane, Piscataway, New Jersey 08854.

Follow IEEEXtreme on

IEEEXtreme 9.0
Student and Proctor Guide

www.ieee.org/Xtreme | IEEEXtreme@ieee.org

Guide Contents:

What is IEEEXtreme?

IEEEXtreme is IEEE's premier programming competition, bringing thousands of students from around the world together into an exceptional 24 hour event. IEEEXtreme is open to all undergraduate and graduate college students who are interested in becoming IEEE student members. The competition is hosted virtually and simultaneously around the world, and competitors are required to be proctored by a local IEEE member and are often supported by a local IEEE Student Branch.

Where can I connect?

- Facebook: www.facebook.com/IEEEXtreme
- Twitter: www.twitter.com/ieeextreme
- Google+: http://goo.gl/4pqknl
- Collabratec: https://goo.gl/fZ04HE

IEEEXTREME PROGRAMMING COMPETITION 9.0

Follow IEEEXtreme on reddit

Logistics and Communications Checklist

Proctors and students should consider the following best-practices when planning and hosting a local IEEEXtreme event:

- ☐ Coordinate with local IEEE Branch Chair & Councilor and university staff on advertising, signups, and facilities for event
- ☐ Circulate resources in official press kit within your local university or college, advertising event with local news sources, student affairs communications, and alumni relations resources
- ☐ Identify proctor to supervise teams—it is often best to find more than one proctor if possible, to help divide the time over the 24 hours
- ☐ Register team members—with an active IEEE student or graduate student membership number—at IEEEXtreme website
- ☐ Review competition rules for proctors' and students' familiarity with procedures, scoring, and prizes
- ☐ Check IEEEXtreme website, Facebook page, and HackerRank platform regularly in days before and during competition for important updates

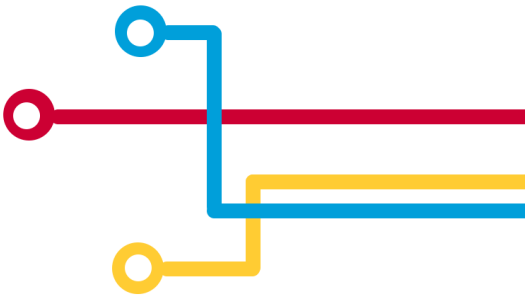
IEEEXTREME PROGRAMMING COMPETITION 9.0

Follow IEEEXtreme on reddit

Facilities Checklist

Proctors and students should consider the following best-practices when planning and hosting a local IEEEXtreme event:

- ☐ Identify and reserve a facility with computers (whether the host's or student's computers) and reliable internet for the full 24 hours of competition—example venues are an IEEE Student Branch office, a college lab, or an other agreed upon location
- ☐ Prepare facilities that will be comfortable and safe—plan for food, water, and exercise needs during the 24 hours
- ☐ For the facility that you are using for the 24 hours, check for any special access accommodations that need to be made ahead of time and assure all participants are aware of any restrictions to their use of the facility
- ☐ Account for transportation needs to and from a local IEEEXtreme event



Follow IEEEXtreme on

Guidelines for Proctors

Each IEEEXtreme team must have at least one proctor to supervise their 24 hours of participation. The following are important points for proctors:

- Proctors must be an IEEE Member of higher membership grade—this means a competing student cannot be a proctor
- Each team can be up to 3 IEEE student or graduate student members, but can only include a maximum of 2 graduate student members per team
- A single school can have multiple teams. A single proctor can support up to 8 teams total at a single location
- The intent and spirit of the competition is for only the students within a team, not others, to solve a problem; in no case will work-on-behalf of teams or individuals be allowed
- It is wise to arrange a couple of proctors for the 24 hours time span
- Although it is not at all required, it is common for proctors to assist students in arranging for food, drink, and sleep needs
- All proctors and students are beholden to the IEEE Code of Conduct

The Problems

IEEEXTREME PROGRAMMING COMPETITION

Please note that the release of the questions will be staggered throughout the contest according to the schedule below*. Problem difficulty levels are subjective estimates, chosen from the ordered list of: easy, moderate, difficult, advanced, or expert.

Hours into competition	Amount of problems released	Difficulty
00:00 - 1:00	5	1 of each difficulty
2:00 - 6:00	5	two(2) easy, one(1) moderate, one(1) difficult, one(1) advanced
07:00 - 11:00	5	one(1) easy, two(2) moderate, one(1) difficult, one(1) expert
12:00 - 13:00	4	one(1) easy, one(1) moderate, one(1) difficult, one(1) advanced
14:00 - 15:00	6	two(2) easy, one(1) moderate, one(1) difficult, two(2) advanced
16:00 - 18:00	4	two(2) moderate, one(1) difficult, one(1) advanced

* The scheduled release of problems as outlined above may be subject to change.

FOLLOW IEEEXtreme 9.0

IEEExtreme9.0

Challenges

Digit Fun!

Success Rate: 63.39% Max Score: 49 Difficulty: Easy

Solve Challenge

Prediction Games

Success Rate: 13.48% Max Score: 89 Difficulty: Moderate

Solve Challenge

Block Art

Success Rate: 14.32% Max Score: 89 Difficulty: Difficult

Solve Challenge

Xtreme Driving

Success Rate: 9.69% Max Score: 92 Difficulty: Advanced

Solve Challenge

Xtreme In Security

Success Rate: 0.14% Max Score: 100 Difficulty: Expert

Solve Challenge

Taco Stand

In progress.

Success Rate: 30.62% Max Score: 76 Difficulty: Moderate

Solve Challenge

Bon Voyage (or Muddle over Cruise Cabins)

In progress.

Success Rate: 1.02% Max Score: 99 Difficulty: Advanced

Solve Challenge

Palindromic Moments

In progress.

Success Rate: 7.33% Max Score: 94 Difficulty: Easy

Solve Challenge

Mr. Pippo's Pizza

In progress.

Success Rate: 35.00% Max Score: 72 Difficulty: Difficult

Solve Challenge

Zoom In

Current Rank: N/A

Message Center

Finite Domain Constraints: Please note that the maximum number of variables in this problem is 10. The problem definition has been updated to reflect this. a few seconds ago

The contest has ended. Further submissions will not affect the leaderboard. 15 hours ago

Current Leaderboard

Review Submissions

In progress.

Success Rate: 45.17% Max Score: 64 Difficulty: Easy

Solve Challenge

High Score

In progress.

Success Rate: 0.28% Max Score: 100 Difficulty: Expert

Solve Challenge

Threesome Poker

In progress.

Success Rate: 25.31% Max Score: 80 Difficulty: Moderate

Solve Challenge

Snakes & Bunnies

In progress.

Success Rate: 15.33% Max Score: 88 Difficulty: Easy

Solve Challenge

IEEE State of Mind

In progress.

Success Rate: 15.02% Max Score: 88 Difficulty: Moderate

Solve Challenge

Car Spark

In progress.

Success Rate: 33.45% Max Score: 73 Difficulty: Difficult

Solve Challenge

Dictionary Strings

In progress.

Success Rate: 29.80% Max Score: 76 Difficulty: Easy

Solve Challenge

Light Gremlins

In progress.

Success Rate: 4.45% Max Score: 96 Difficulty: Moderate

Solve Challenge

Chocolate 2

In progress.

Success Rate: 1.05% Max Score: 99 Difficulty: Difficult

Solve Challenge

SAD: Long Way from Home

In progress.

Success Rate: 0.91% Max Score: 99 Difficulty: Advanced

Solve Challenge

Shortening in the Real World

In progress.

Success Rate: 56.67% Max Score: 55 Difficulty: Easy

Solve Challenge

Communities

In progress.

Success Rate: 21.40% Max Score: 83 Difficulty: Advanced

Solve Challenge

Pattern 3

In progress.

Success Rate: 15.71% Max Score: 87 Difficulty: Easy

Solve Challenge

Land

In progress.

Success Rate: 0.72% Max Score: 99 Difficulty: Difficult

Solve Challenge

That's So Characteristically Euler!

In progress.

Success Rate: 20.29% Max Score: 84 Difficulty: Moderate

Solve Challenge

Finite Domain Constraints

In progress.

Success Rate: 0.98% Max Score: 99 Difficulty: Advanced

Solve Challenge

Blocks Game

In progress.

Success Rate: 10.05% Max Score: 92 Difficulty: Moderate

Solve Challenge

Would Be... Could Be... BST!

In progress.

Success Rate: 0.76% Max Score: 99 Difficulty: Advanced

Solve Challenge

Alice and Bob Play Sheldon's Favorite Game

In progress.

Success Rate: 15.06% Max Score: 88 Difficulty: Moderate

Solve Challenge

Eat, Sleep, Drink, Code

In progress.

Success Rate: 2.11% Max Score: 98 Difficulty: Difficult

Solve Challenge

#1

Digit fun!

There are ten digits, You should know, And we will sing them, Here we go, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

Digit Fun!

HackerRank

Problem Statement

Recurrence relations are an important tool for the computer scientist. Many algorithms, particularly those that use divide and conquer, have time complexities best modeled by recurrence relations. A recurrence relation allows us to recursively define a sequence of values by defining the n^{th} value in terms of certain of its predecessors.

Many natural functions, such as factorials and the Fibonacci sequence, can easily be expressed as recurrences. The function of interest for this problem is described below.

Let $|A_n|$ denote the number of digits in the decimal representation of A_n. Given any number A_0, we define a sequence using the following recurrence:

$A_i = |A_{i-1}|$ for $i > 0$

The goal of this problem is to determine the smallest positive i such that $A_i = A_{i-1}$.

Input Format

Input consists of multiple lines, each terminated by an end-of-line character. Each line (except the last) contains a value for A_0, where each value is non-negative and no more than a million digits. The last line of input contains the word END.

Output Format

For each value of A_0 given in the input, the program should output one line containing the smallest positive i such that $A_i = A_{i-1}$.

Sample Input

```
9999
0
1
9999999999
END
```

Sample Output

```
3
2
1
4
```

Explanation

The first input value is $A_0 = 9999$, resulting in $A_1 = |9999| = 4$. Because 4 does not equal 9999, we find $A_2 = |A_1| = |4| = 1$. Since 1 is not equal to 4, we find $A_3 = |A_2| = |1| = 1$. A_3 is equal to A_2, making 3 the smallest positive i such that $A_i = A_{i-1}$.

The second input value is $A_0 = 0$, resulting in $A_1 = |0| = 1$. Because 0 does not equal 1, we find $A_2 = |A_1| = |1| = 1$. A_2 is equal to A_1, making 2 the smallest positive i such that $A_i = A_{i-1}$.

The third input value is $A_0 = 1$, resulting in $A_1 = |1| = 1$. A_1 is equal to A_0, making 1 the smallest positive i such that $A_i = A_{i-1}$.

The last input value is $A_0 = 9999999999$, resulting in $A_1 = |9999999999| = 10$. Because 10 does not equal 9999999999, we find $A_2 = |A_1| = |10| = 2$. Since 2 is not equal to 10, we find $A_3 = |A_2| = |2| = 1$. Since 1 is not equal to 2, we find $A_4 = |A_3| = |1| = 1$. A_4 is equal to A_3, making 4 the smallest positive i such that $A_i = A_{i-1}$.

#2

Prediction Games

You will be hungry again in one hour! I see pizza delivery coming.

Prediction Games

Problem Statement

In a *Prediction Game*, two or more players attempt to predict the score of a series of upcoming sporting competitions. Each player's predictions are then evaluated based on the sum of the categories listed below. These descriptions use the following variables:

- S_1: Actual score earned by team 1.
- S_2: Actual score earned by team 2.
- P_1: A player's predicted score for team 1.
- P_2: A player's predicted score for team 2.

Point System

- *Winner*: 10 points if they correctly picked the winner, i.e. the team with more points at the end of the competition.
- *Team 1 Score*: Maximum(0, 5 - $|S_1 - P_1|$). In other words, if they pick the correct score for Team 1, they earn 5 points. If they were off by one in their prediction, they earn 4 points. If they are off by two, they earn 3 points, and so on. If they are off by 5 or more points, they earn 0 points in this category.
- *Team 2 Score*: Maximum(0, 5 - $|S_2 - P_2|$).
- *Point Spread*: Maximum(0, 5 - $|(P_1 - P_2) - (S_1 - S_2)|$). In other words, if the player predicts the correct number of points by which one team beats another, they earn 5 points. If they are off by one, they are 4 points, and so on. If they are off by 5 or more points, they earn 0 points in this category.

Some, but perhaps not all, of the competition scores have been received. Your task is to provide a list of players who might end up with the highest point total when all scores are recorded.

Notes:

- Whenever we refer to a "game," we are referring to the "Prediction Game." When we refer to a competition, we mean the sporting event that teams are playing.
- No competition will end in a tie. In addition, the players are not allowed to predict a tie score as a final result for a competition.
- The smallest score possible in a competition is 0 points. The maximum possible score in a competition is 200 points.
- A player can earn points in the *Team 1 Score*, *Team 2 Score*, and *Point Spread* categories, even if they do not correctly pick the winner. For example, if a player predicts a score of 24-17 (i.e. 24 points for team 1 and 17 points for team 2) and the final result is 23-30, the player would earn 4 points for the *Team 1 Score* category (since the difference between the prediction (24) and the true score (23) is only one, he would get 5-1=4 points). He would receive no points for the *Team 2 Score* category or for the *Point Spread* since they are both off by more than 5 points. Similarly, if a player predicted a score of 10-9 and the actual score was 9-10, the player would earn a total of 11 points as follows: 4 points for *Team 1 Score* (again 5-1=4 points since the difference between the predicted and the actual score is 1), 4 points for *Team 2 Score* (for the same reason as with scoring analysis for Team 1), and 3 points for

the *Point Spread* (since according to the formula the point spread is |(P1 - P2) - (S1 - S2)| = |(9-10) - (10-9)|=|-1-1|=|-2|=2, and thus the score obtained for this prediction should equal 5 - *Spread* = 5 - 2 = 3 points).

Input Format

The first line of input contains an integer t, $1 <= t <= 20$, indicating how many test cases are in the input.

Each test case follows in the following format. First comes a line with two space separated integers, p and c, where p indicates how many players there are, and c indicates how many competitions are being played.

The following is repeated for each of the p players. The first line contains the player's name. Then follows c lines with two space separated integers. These are the predictions for this player. The first of these prediction lines contains P_1 and P_2 for the first competition. The second contains the predicted scores for the second competition, and so on.

Finally there are c lines indicating the (possibly) partial results for the competition. The first competition's result is reported on the first of these lines, the second competition's on the second of these lines, and so on. Each of these lines will contain two space separated values. If the score for the team is known, the values will be integers giving the score. If the result of the competition has not yet reported, the line will read `? ?`.

Notes:

- Each player's name is unique. All of the names are comprised of an initial uppercase letter, followed by up to 30 lowercase letters.
- $2 <= p <= 20$
- $1 <= c <= 1000$
- Either both scores will be given or neither will be given. You will never be given a competition result where only one of the scores has not yet been reported.

Output Format

For each of the test cases, you should output a line containing, in alphabetical order, a space-separated list of the players who could earn the maximum number of points. If there is only one winner possible, then only the winner's name should be reported on a line of its own.

Sample Input

```
3
2 3
Alice
14 17
20 7
30 7
Bob
20 7
21 17
14 13
14 17
17 13
? ?
2 3
Dave
14 17
20 17
30 7
Chuck
20 10
```

```
27 17
30 7
? ?
27 17
30 7
3 1
Francis
14 7
Eve
10 21
George
7 30
0 1
```

Sample Output

```
Alice
Chuck Dave
Eve George
```

Explanation

There are three test cases.

Test Case 1

In the first competition, Alice predicted a score of 14-17, which was also the actual score. She earned 25 points for this prediction (10 points for the winner and 5 points in each of the other categories).

In the second competition, Alice predicted a score of 20-7, and the actual score was 17-13. She earned 12 points for this prediction, 10 points for the correct winner and 2 points for the score prediction for team 1 [5 - |20 - 17| = 2].

In the first competition, Bob predicted a score of 20-7, and the actual score was 14-17. He earned 0 points for this competition, because he was wrong about the winner, and he was off by more than 5 on both team scores and on the point spread.

In the second competition, Bob predicted a score of 21-17, and the actual score was 17-13. He earns 17 points for this prediction (10 points for picking the winner, 1 point for team 1 score, 1 point for team 2 score, and 5 points for correctly predicting the spread).

With these known results, Alice is leading Bob, 37 points to 17 points. Since both Alice and Bob picked the same winner for the third competition, there is no way for Bob to catch up. Therefore you would output just "Alice" for this test case.

Test Case 2

In this test case, Dave currently has 40 points: 15 points from the second competition (10 points for picking the winner and 5 points for picking team 2's score), and 25 points for picking the correct score for the third competition. Chuck has 50 points for picking the correct scores in the second and third competitions.

Suppose the result of the first competition was 10-17. Then, Dave would end with 57 points and Chuck would end with 50 points.

On the other hand, suppose the result was 30-0. Then Chuck would end with 60 points and Dave would end with 40.

Therefore, since either Chuck or Dave could end with the highest number of points in the game, you would output "Chuck Dave" (since "Chuck" comes before "Dave" in alphabetical order).

Test Case 3

Here all of the results are in, and Eve and George have the high score of 10 points each.

Inspirational Quote

Here's a quote to reward you for reading the entire problem, and inspire you to solve it!

"It's tough to make predictions, especially about the future." - Yogi Berra

#3

Block Art

The Journey from Cubism to NeoCubism: From Picasso to Blockhead.

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Block Art

Problem Statement

The NeoCubist artistic movement has a very distinctive approach to art. It starts with a rectangle which is divided into a number of squares. Then multiple rounds of layering and scraping occur. In a layering round, a rectangular region of this canvas is selected and a layer of cubes, 1 cube deep, is added to the region. In a scraping round, a rectangular region of the canvas is selected, and a layer of cubes, again 1 cube deep, is removed.

The famous artist I.M. Blockhead seeks your help during the creation of this artwork. As he is creating his art, he is curious to know how many blocks are in different regions of the canvas.

Your task is to write a program that tracks the creation of a work of Pseudo-Cubist art, and answers I.M.'s periodic queries about the number of blocks that are on the canvas. Consider, for example, the following artwork created on a canvas with 5 rows and 15 columns or squares. The canvas starts without any blocks, like in the figure below. We label cells in the canvas based on the tuple (row,column), with the upper left corner being designated (1,1). The numbers in each cell represent the height of the blocks in that cell.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After adding a layer in blocks to the rectangle with upper left corner at (2,3) and a lower right corner of (4, 10), the canvas now looks like the following:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After adding a layer of blocks in the rectangle with upper left corner at (3,8) and a lower right corner of (5, 15), the canvas now looks like the following:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	2	2	2	1	1	1	1	1
0	0	1	1	1	1	1	2	2	2	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

If Blockhead were to ask how many blocks are currently in the artwork in the rectangle with upper left corner (1,1) and lower right corner (5,15), you would tell him 48.

Now, if we remove a layer of blocks from the rectangle with upper left corner at (3,6) and a lower right

corner of (4, 12), the canvas now looks like the following:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	1	1	1	0	0	1	1	1
0	0	1	1	1	0	0	1	1	1	0	0	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

If Blockhead were to ask how many blocks are now in the artwork in the rectangle with upper left corner (3,5) and lower right corner (4,13), you would tell him 10.

“Beautiful!” exclaims Blockhead.

Input Format

The first line in each test case are two integers r and c, $1 <= r <= 12$, $1 <= c <= 10^6$, where r is the number of rows and c is the number of columns in the canvas.

The next line of input contains an integer n, $1 <= n <= 10^4$.

The following n lines of input contain operations and queries done on the initially empty canvas. The operations will be in the following format:

[operation] [top left row] [top left column] [bottom right row] [bottom right column]

[operation] is a character, either “a” when a layer of blocks is being added, “r” when a layer of blocks is being removed, and “q” when Blockhead is asking you for the number of blocks in a region.

The remaining values on the line correspond to the top left and bottom right corners of the rectangle.

Note: You will never be asked to remove a block from a cell that has no blocks in it.

Output Format

For each “q” operation in the input, you should output, on a line by itself, the number of blocks in the region of interest.

Sample Input

```
5 15
5
a 2 3 4 10
a 3 8 5 15
q 1 1 5 15
r 3 6 4 12
q 3 5 4 13
```

Sample Output

```
48
10
```

Explanation

This test case corresponds to the description given above.

#4

Xtreme Driving

Please, cows, moooove out of the way!

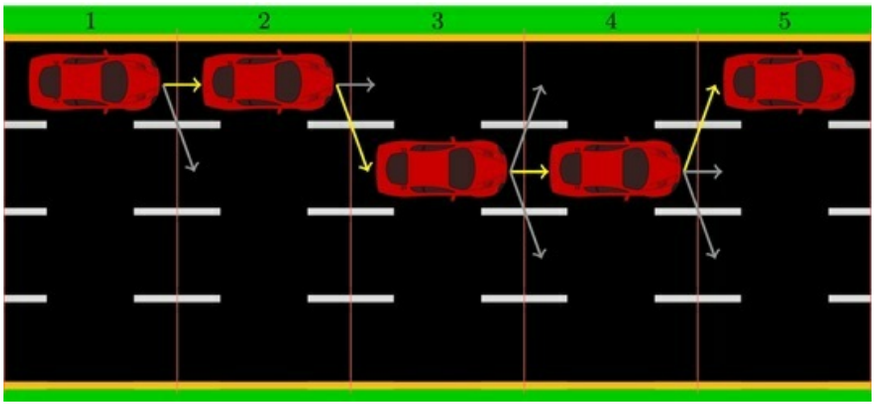
Xtreme Driving

Problem Statement

Alice, one of the IEEEXtreme participants, is on her way to her university to take part in this year's contest. To get to the university she has to drive on a four lane highway, but as the highway is very long she quickly becomes bored. She decides to practice for the contest by thinking about some problems related to the highway she's driving on. She comes up with the following problem:

Let's say that, in a single unit of time, her car, which is of unit-length, can perform one of the following three actions:

- Drive one unit forward, staying in the same lane
- If the car is not on the left-most lane, drive one unit forward and switch to the lane on the left
- If the car is not on the right-most lane, drive one unit forward and switch to the lane on the right



If the highway is K units in length, in how many ways is it possible to drive through the highway, provided that she starts on the first unit of the highway at the left-most lane, and ends at the last unit of the highway, also at the left-most lane?

As she's been training very hard for the contest, it doesn't take her long to come up with a solution to this problem. But all of a sudden, out of nowhere, a large cow appears in front of her car, and she just barely manages to avoid crashing into it. She got a bit too distracted thinking about the problem... But this incident gives her an idea: what if the highway had a set of stationary cows that the car must avoid crashing into?

Unfortunately she doesn't have time to think about this version of the problem as she just arrived at the university and the contest is about to start. When the contest starts, she is very surprised to see that the problem she was thinking about is just like one of the problems presented in the contest. What a coincidence! Again her hard practice pays off and she quickly solves the problem. But the real question is, can you?

Input Format

Input begins with two integers, K, the length of the highway, and N, the number of cows standing on the highway, subject to the following constraints:

$2 <= K <= 10^{18}$

$0 <= N <= 100$

$N <= 4(K - 2)$

Then follow N lines. The i^{th} line contains two integers describing the location of the i^{th} cow, x_i, the lane on which the cow is standing (1 being the left-most, and 4 being the right-most), and y_i, the unit on which the cow is standing, subject to the following constraints:

$1 <= x_i <= 4$

$1 < y_i < K$

Notice that no cows are on the first or last unit of the highway. It is also guaranteed that no two cows share the same position.

Output Format

You are to output the number of ways to drive from the left-most lane at the first unit of the highway to the left-most lane at the K^{th} unit of the highway subject to the rules described above, and the additional constraint that the car must not drive to a position occupied by a cow.

As the number of ways can be quite large, please output the answer modulo $10^9 + 7$. Note that this is the same as the remainder when dividing the number of ways by $10^9 + 7$.

Sample Input

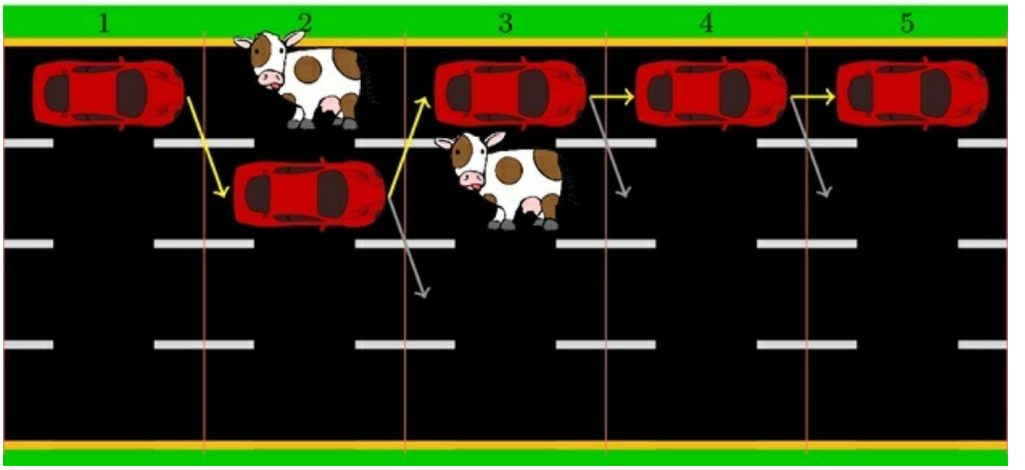
```
5 2
1 2
2 3
```

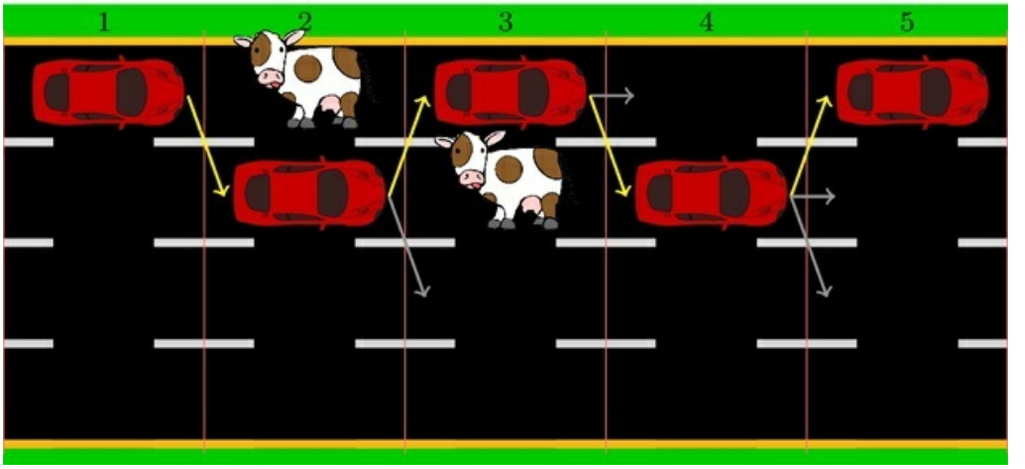
Sample Output

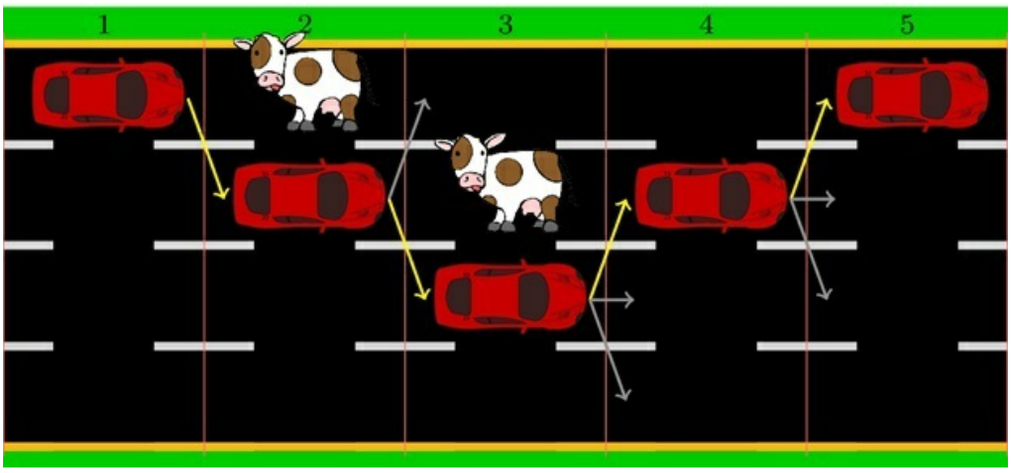
```
3
```

Explanation

In this example there are three ways to drive through the highway, and they are shown in the following figures. Notice that the car can drive in between cows (as long as other driving rules are fulfilled), but it must not drive to a unit occupied by a cow.







Note: Two more sample test cases are available if you click on the "Run" button.

#5

Xtreme In Security

Bad passwords and worse crypto

IEEEXTREME PROGRAMMING COMPETITION 9.0

ELECTRON DEVICES SOCIETY

IEEE

Xtreme In Security

Problem Statement

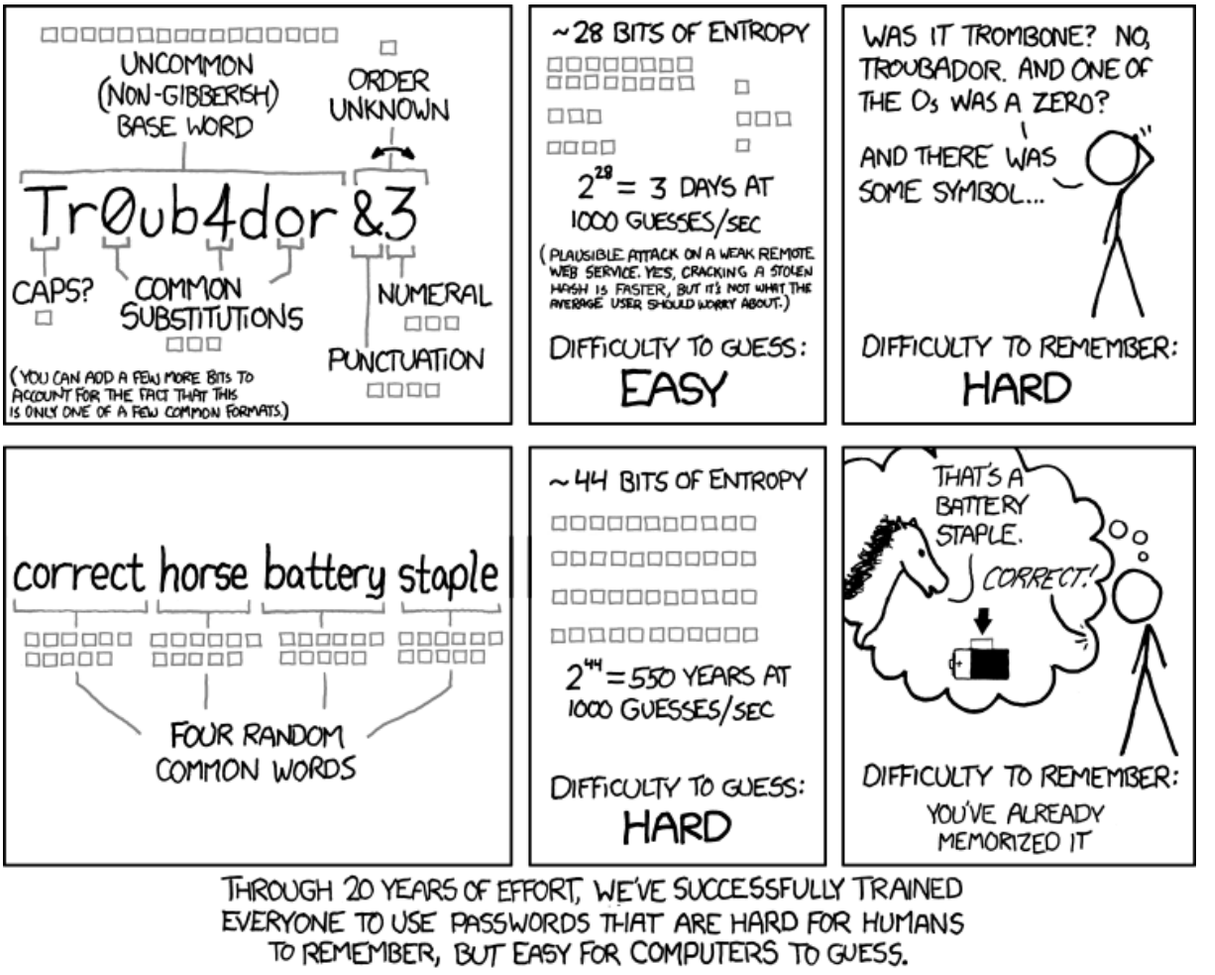
The Xtreme In Security Corp. has devised a password-based authentication system for their new operating system. They have made some unfortunate design decisions that make their system vulnerable to attack.

The system stores in a file a salted and peppered password hash. All of the passwords use the same salt "IEEE" and the same pepper "Xtreme". When the user is enrolled, they supply a password *p*. The system checks to make sure that *p* is less than 20 characters long, and that it contains only lowercase letters and numbers. Then the salt is prepended to the password and the pepper value is appended. The resulting value, using UTF-8 encoding, is then put through the SHA-256 hash algorithm, and then base64 encoded. This base64 encoded value is stored in a password file, along with the user's name.

Note: Even though the password contains only lowercase letters and numbers, the salt and the pepper contain some uppercase letters. The SHA-256 hash algorithm is case sensitive.

When the user is authenticated, a similar process occurs. The user is prompted for a username and a password. Then the salt is prepended to the supplied password and the pepper is appended. The resulting value is hashed with SHA-256, and base64 encoded. If the resulting value matches what is stored in the password file for the specified user, then the authentication is successful.

Your task is to break as many of the hashed passwords as possible to try to convince the system designers to improve their approach. Here is some inspiration from XKCD:



Credits: XKCD

Input Format

The input will consist of a single salted, peppered password hash. This hash will be derived from a single password.

The hash will be chosen from the following set of values:

```
/PtjJboZGlsmTovvyOhBOoTVnQKUP/gJXxjLAW9Lppw=
05HwH93tksb69U1ifesCQuYFP+gKPVH2L6W8JeBdXy0=
0BkyqI3NHyjh0m20wNt6txW08dgISMP4/qzUEezq4Aw=
1mT5cdKRz4BbfMdc8LAdnxfjsGO4lV0k0/V1IHtidmY=
28AdfW0JHmCP4TbieGON8dafRaFpUgpzuX2bHZN6WsM=
3hoie8omUyvM/9Qfx9dKfoptlwemYe2os8aohTGzoyw=
3uDaglldYUn11AadEhELBjE15A0L6hAaWnZmjCGtt+M=
4D4NstlYSjVN826Q+SXUDDmXgIJplpYWiJYf9rt7H8U=
7FCsEXCDTAxyLh3EPnNx7YrJ44SzehQYv3GmPSA6pWk=
81X3NN5JgTuGgqq3ErF0eL/l/wZFYkCwur6fZ06L2Us=
8FzGA/nS7XizLrAVOr/FoeKSq4gaoQRq+kpBKNXHlzc=
8OtpJ+E1XHv2RDsKlEwJc9KUFwPRzaqeHJ735Er6AVE=
8cdgZu9dBOrcTBMqElM+y9Vh5FTBRQ7n9EGa/4qVHUk=
8esDbw8ZVmUuUMEy2Scf5qGiZYiykevrvKpq2bVYHj4=
9DS4orbhPFbjJcosEqQg+eg0Si5qSOnftfHiqK8sYug=
9XDFlu4RPH0EL5XR+5VYILJ3UFAyltpfjONJKp/vcLk=
B2E/K/DywbLEKutOKpS8HxHFrZwucwac4KjzYgsXg3g=
BJQeqlV+4ejv0je5GpekzGdHHWHL5nnrbtD/170LZCA=
BjQiH/A2FUNHlUwhBi8NWmj3HmhmAh6Ag0kRyVSaVo8=
BwbAsOqPsxteVCpAwljrhYogsUS1bF/bLns2QdcLYUI=
CYDZabjeyTAwcEDEcvrX83514UmpjzvQUQ68DIZ/PXg=
CornANxoZ5FJnlhwHmK42CDXf3h6jFr3g73YIRuoymQ=
DDa2TJX20pPsNftfyJ3s6LBwSMSR3EADZGDxW2wThbs=
FFXy3vru2D8rTWZRlh9lSMvtEusfWgO17OmJCTQTECs=
FGkqFC/jLDqDZ10fqI1nGw7AQNWioOrZ3ydEaJyXBwo=
Fz+Y0H/R2rMZlc1C88Yx0A0xluYnVTinlw5qaSx8vWQ=
GcJMWMDF6+f+onf6oi1FbpnN7dVrFEZnlXtHqmaygs4=
GsXTQM0w+Clb1c9B7n28mADU2quLeI1n91KTyBboeHl=
HLnuqQmCYetzrau3frCDEpZ52QCIby108gugsmwAwQ0=
HWv9gx+GL/6g+0b0eOc+1Z/8BHse91/5T/DdiDwEknU=
IDB/pOthrWobzapJ/N8HsraNhwfbExAa2uusdiKHFFI=
IXYqlHbVeONERbxFe8SaEPEEKex45EihiC/l8CR52kk=
Idvs4Al9YZsqPG8xkSxVqb6MOVhbw5k+qtF8UZKYVE8=
IxQxcFXR51q8h8FLblPhYfUR30llAt6hX8TjZWVa/GE=
JCmqBN0MsW13tEmsQPYWg9Fj25MUrqFYvSK2arxTt0A=
JOXxLH/i8i+fxDlWP2cts5Si/5En1A8M3s/vy6Aadic=
KudA8vCEQdGaaCSxotpAcluXnVPS3MAZPkwl/lVupak=
LGZEfbUr/tMREpJtsao/uuewcJXApmgfHDbh1zzfdhU=
LZIzmWEqXDPJsnKttFGRaG/jUhHrbTEKt1XCO6XbdME=
Lq0kV5M0HDSgB4m5KZbbn6BYRNlkKfaaAr3/11ueopY=
Lqxt1UjT1ecV6ucgYn+yrGSUTxPWkZgdDtbygGwC/BA=
MHQTB1MSsvhBxMpdRUiaM9Uj1QxU7zYq3FqDlW2HT2s=
MKewBZryb2l36Y0tDyx+WuVeXUGfiSzcJplm9y1w9m0=
N5aunKGHeN2WETLXLzfhfCxAfkwtGU/imziiTF3t7oY=
NmrOUzHxKSfNT8UJ1YXbRL8I+HCAb+gIJ0bBXcHfagE=
NnSS+AuW1Z6zytSfqailVp4xxHHe70Av+IdhDlkoltQ=
O9L1ZWYwuzgaImTjOwuogXfpC+C44zzcDhpt08LjR5c=
R/ye+L9W+l6hyZ/v1POsWYboEGemlisARL8ohUvfBLI=
R1v9fEb9VrZuU5xiYTKTqhHF03VtXg7+KtfFHPkQuCQ=
RVfhsLovxa+/6tWgeSBASIlkzXkVtDPT4yYvjboHhlk=
Rz38Ng2qI3mPkaRB6uDoCYmmfzbVTCzpt2sG1o+TZqo=
S98FBzlv2vMVP/q+23m1wrHMJlrcy1rhoQGy338c4Bs=
SzraQWWasG5ZO1tJq16DqU/7M/o/WRiAWRl1aFFvwr8=
TInfNYwXvofBA+9QIe3+XEfDpO5ER+R+Jn3BOshhZWU=
VDkcRd54BhYlK5Wg2PPDa/jzGrSkMepGIv6Tw3I2ksc=
VjFTqTEY27V8lK2yCvhLhYm2Brh9bN1vWckVaevsiiY=
Wau4ReopjFKk8SYYNq5llBL+Rgg8aBR6h9UgTlv2u7I=
XtEWsXf16y6Bc7vQxDy7hwRdBVWo3dV9C6CDVSf4PLs=
Y5b6UztMueFYIFMl4a61jlD/ZhFG74/rVn8XaqqU+8c=
YollqBlewcxE/kF6PKvv0r1CLZkKx82657bB0eQbiK0=
YzSqlQTtq5j+Kd+hW1lSgBW0mn61vsQsxNeipq0sYCo=
aJlH+q0YjYZCpierKtbue5JDtZSF8tKxVKuHYUPQ65k=
bi6rh2HgTbJxR2GOTNWZLlxiiWZVnObptGj0KqOCSYo=
cX7VyMvSYhuRvEfAUb3uNh8kmjpNFg0tatUPN562iOg=
eSCTiCrzHPbngPu3F4ivPkLUv7MqLUlmWAhA4UO975Y=
eUqkcVCbgIx1bGhhmnN5MxJFJhVruINmG65TjT/EQ1k=
gE7PseB3mspPtYG3JROzT9FeqTfPFYQvBF2SJD9V19Y=
gMi4hC4o7Fmv6ylrU48BVy8I1khXwkaD36G7bWiZHeo=
h84yifAWGLj9sakEqxZ3QEjkXL42AoScP3L5Tdevm98=
```

```
hEGKCiTZSA5x560hodRoIBBTE8pv4sP1VXG4D0fXWcI=
ix+0IJlpLpHeSHEll+Q/IVY+FlRXn3xMA0ey6UlTi34=
j2GTUqtqZpotY8wF16zkvnbdCLTnX3oOZ30SjQUnIUk=
jtUg00EsmzzFkk8JgKg3OpkmPRpN9xbwsdNXQSPczwo=
k1J9Dv3El442CO6A2FGN1H8JgFO2kjNBvkjDR0WlvkA=
kuRpklh9kaNz6XvG8U6GO/IARH/SqhRnTiJbZHXC0yQ=
lFgqRrqTz1WXmO22u+ls1ZmWWUtuYrTJigsSB7I9NHI=
lUfxHX9xH2aOHheMMqQF+f5BNh97avew2uOwEN3B7HE=
m0IeuugWDOl9cFUFRYJhouCBT39T0dpp1xBOKPqHP94=
mgA5kgALstQpGUBp4vZ8oiz0P4jjAGl0wjgls6kQyMA=
nMAwaDYvElAwoqtqWMpBAWdhuRgRq/fmwWbRM7cOMlU=
opBtoC66YDRbLNqZAAu2FleKfF0HMOGHCOCPYeNHdx4=
ot/igM+me4e3UTT731qcBkSAcToyADMCddr7i7LCWGo=
qrkd/8imuRtiDb9N1w2hQRxJAkdx3Wqh1HVXPS7dym8=
r6BN0tdyAYZD0Nmc7bfV0WRcFBb1A2WlPPKHVRG59k8=
rdALvOYVhA3hnUTBlaQXigWBSgMYzGTreSKyMoAoKfw=
rjwtKqkPc7cfQ+zZ+E9c+fzQYhRvhVtaKEFb+srlHcQ=
rwvmTDiJxlEETbsngvpxYGwZZK+FGo7527odGuQUjtQ=
socJeO02bT2w+XZUrLoopbZvQ1lRhDfE88GVrJQ8p+g=
tDdmKQpMiVDFA1YdblkHSFzL4Z9UlQ9FSouf3TybOu0=
tojYiNtlWmq+7r1dSvxDk3W5at/NMAi1uxCHY61WAKk=
tqpGCBzhR+0ONFk1sBiHPhz+kRiXmY3CGdUXqnMJwLg=
uAZthS7b4ySZtWpM9pJ7ulYnhFdpFABpR2iPRQEmff0=
uCG9dSBejCOrZWsX7+u8G340p74s8lDS/El8MIeOyMo=
ugclIpDID0R1uFqBAcN3PNXhwlhen77GdAccFgpbs+A=
usg8BTtSfewL5M3OVg91TJCTc5vONLqgUCC/Si1Grsg=
vs2sCU8qG0pDYQfcjlPzDzvcbJnhP1OgFRcXP4i3ffw=
wEtqAs8JHjicWnXshAWF5Sg6NoswuG9qeJ7USw7YD7c=
y+zbMpySKY+WF97KkgRQ+tBpM7iTqTj9guWmGJcqfyA=
y1R6JQiUzUovgtdrvCkbeQAyMhFoupzhI5ZuQVPfCgw=
zqKPAOt5ziHSeRxc0TgUZF3rJxzBHAKdJeccvt3F7Jg=
```

Output Format

The output should consist of a single alphanumeric string, which hashes to the corresponding value given in the input.

Note: You may not be able to break all of the hashed passwords. If you are given a hash that you cannot break, you may output an arbitrary string. You will earn partial credit for this problem for each password that you do crack!

Sample Input

```
tDdmKQpMiVDFA1YdblkHSFzL4Z9UlQ9FSouf3TybOu0=
```

Sample Output

```
password1
```

Explanation

For all passwords, the salt value is "IEEE" and the pepper is "Xtreme". First, we can take the SHA-256 hash of the UTF-8 encoded string `IEEEpassword1Xtreme`. If we then base64 encode the result, the value equals the desired value: `tDdmKQpMiVDFA1YdblkHSFzL4Z9UIQ9FSouf3TybOu0=`.

#6

Taco Stand

Help Joe sell some tacos!

Taco Stand

Problem Statement

Joe has been hired to make tacos at a series of baseball games. He wants to calculate the maximum number of tacos he can make based on the available ingredients. He always insists on fresh ingredients, so any leftover ingredients on a given day will be thrown away.

His ingredients are:

- Taco shells - every taco gets exactly one of these
- Meat
- Rice
- Beans

His recipe is to take one taco shell, then add exactly two of the ingredients: meat, rice, and beans. So, for example, one taco might have meat and rice, while another taco might be made with rice and beans. However, a taco *cannot* have two of the same ingredient. For example, Joe will never make a taco with two servings of meat.

Your task is to write a program to calculate the maximum number of tacos Joe can make each day, given the amount of ingredients he will have.

Input Format

The first line of input is an integer n, $1 <= n <= 1000$, specifying how many days Joe will be making tacos.

The following n lines contain 4 space-separated integers in the format:

$s\ m\ r\ b$

where s is the number of shells available, m is the amount of meat, r is the amount of rice, and b is the amount of beans, each expressed in terms of the number of tacos they could make.

Note: s, m, r, and b are all non-negative integers less than 10^9.

Output Format

The output file is exactly n lines long, each line containing an integer specifying the maximum number of tacos Joe can make with that day's ingredients.

Note: There is a newline character at the end of the last line of the output.

Sample Input

```
2
5 3 4 1
1 9 9 9
```

Sample Output

```
4
1
```

Explanation

On the first day, Joe can make a total of 4 tacos - 3 meat and rice tacos and 1 rice and bean taco.

On the second day, Joe only has one shell, so he can make 1 taco with any two of the ingredients.

#7

Bon Voyage

(or Muddle over Cruise Cabins)

Sail away with our Cruisetanic

Bon Voyage (or Muddle over Cruise Cabins)

Problem Statement

The TransContinental Ocean-liner Company is celebrating their centennial anniversary. As a part of the celebrations, they launched an exciting contest for a free, around-the-world cruise on one of their ships. As part of the contest entry, passengers select two cabins in which they would like to stay. If they are a winner, they will be given one of the two cabins.

Interest was enormous, and great many people applied to be a part of the voyage. Winners will be decided on a lottery. The company does not have any control over whom all will win the final lottery or obtain the tickets, but only on how many tickets will be distributed.

Jerry, the tour manager, is finding it difficult to find the maximum number of prizes that can be awarded. He must make sure that each potential winner can be accommodated in one of the cabins that they specified when they entered the contest.

Your job is to find the maximum number of prizes that can be awarded subject to this constraint. Help Jerry to find a solution for his problem and earn some real good points for yourselves!

Input Format

The input begins with an integer T, $1 <= T <= 15$, indicated the number of test cases.

Each test case consists of the following:

- A line containing an integer N, $2 <= N <= 200$, where N is the total number of rooms on the ship.
- A line containing an integer M, $0 <= M <= 500$, where M are the number of people who signed up.
- Next come M lines, the i^{th} line containing the pair of cabins selected by the i^{th} entrant in the contest. The cabin choices will be two space-separated integers, each falling between 1 and N (inclusive).

Output Format

For each test case, you should output, on a line by itself, the maximum number of tickets that can be awarded such that every winner can be housed in one of the two cabins that they have selected.

Sample Input

```
5
6
4
1 2
1 2
3 4
5 6
6
5
1 2
1 2
1 2
3 4
5 6
4
```

```
5
1 2
1 3
1 2
2 3
3 4
5
11
1 2
2 3
3 4
4 5
1 5
2 4
3 4
3 5
2 5
1 2
3 5
3
6
1 2
1 2
1 3
2 3
1 2
1 3
```

Sample Output

```
4
2
3
3
2
```

Explanation

The sample input consists of 5 test cases.

In the 2nd test case, 3 people opted for the same 1 or 2 cabins, and in order to prevent the worst case of all the 3 of them getting the ticket, we can give out at most 2 tickets.

In the 3th test case, we can give out 3 tickets and still make sure that regardless of who gets a ticket, any winner can be housed in a cabin of their choice.

#8

→ EVIL OLIVE ←

Palindromic Moment

Are we not drawn onward, we few, drawn onward to new era?

IEEEXTREME PROGRAMMING COMPETITION 9.0

MORGAN & CLAYPOOL PUBLISHERS

IEEE

Palindromic Moments

Problem Statement

Bob, Hannah, and Otto like to celebrate palindromic dates. A palindromic date is one in which the numbers in the date read the same forwards and backwards. They were very happy with 2015, which has 15 palindromic dates:

- 5th of January, October, November and December in `day/month/year` format: 5/1/15, 5/10/15, 5/11/15, 5/12/15
- Eleven days in May listed in `month/day/year` format: 5/1/15 and 5/10/15 through 5/19/15

Note: These dates are palindromic in the sense that, if the forward slash delimiter (/) is disregarded, the numbers in the date read the same forwards and backwards.

While they wait to celebrate the next palindromic date on November 5th, they have decided to they would like to start celebrating palindromic moments. They define a palindromic moment as a palindromic string obtained by formatting a date-time combination using one of the following date prefixes and one of the following time suffixes:

- Date prefix:

  Combinations of month, day, year: `Mdyy`, `MMdyy`, `Mddyy`, `MMddyy`, `Mdyyyy`, `MMdyyyy`, `Mddyyyy`, `MMddyyyy`

  Combinations of day, month, year: `dMyy`, `dMMyy`, `ddMyy`, `ddMMyy`, `dMyyyy`, `dMMyyyy`, `ddMyyyy`, `ddMMyyyy`

- Time suffix:

  Combinations of hour, minutes, seconds: `hmmss`, `hhmmss`, `Hmmss`, `HHmmss`

Where

- `M` is the month of the year as a number, e.g. January = “1” and October = “10”
- `MM` is the month of the year as a number with a leading zero if the month of the year is one digit long, e.g. January = “01” and October = “10”
- `d` is the day of the month as a number, e.g. first = “1” and fifteenth = “15”
- `dd` is the day of the month as a number with a leading zero if the day of the month is one digit long, e.g. first = “01” and fifteenth = “15”
- `yy` is the last two digits of the year, e.g. 2000 = “00”, 12015 = “15”
- `yyyy` is the complete year as a number, e.g. 512 = “512”, 2000 = “2000”, 12015 = “12015”
- `h` is the hour in (AM/PM), e.g. 1 pm = “1”, 12 am = “12”
- `hh` is the hour in (AM/PM) with a leading zero if the hour is one digit long, e.g. 1 pm = “01”, 12 am = “12”
- `H` is the hour in the day, i.e. 0 <= H <= 23, e.g. 1 pm = “13”, 12 am = “0”

- `HH` is the hour in the day with a leading zero if the hour is one digit long, e.g. 1 pm = “13”, 12 am = “00”
- `mm` is the minute in the hour, 5 past the hour = “05”, 30 past the hour = “30”
- `ss` is the second in the minute, 5 past the minute = “05”, 30 past the minute = “30”

For example, during this competition on October 24, 2015, there are four palindromic moments:

- 1:01:42 AM (using the `dMyy` prefix and the `hmmss` suffix): `24101510142`
- 1:42:01 AM (using the `Mdyy` prefix and the `hmmss` suffix): `10241514201`
- 1:01:42 PM (using the `dMyy` prefix and the `hmmss` suffix): `24101510142`
- 1:42:01 PM (using the `Mdyy` prefix and the `hmmss` suffix): `10241514201`

Note that for October 24, the `MMddyy`, `Mddyy`, and `MMdyy` prefixes are equivalent to the `Mdyy` prefix. Similarly for this date, the `ddMMyy`, `ddMyy`, and `dMMyy` prefixes are equivalent to the `dMyy` prefix.

Input Format

The input begins with an integer *t*, 1 <= *t* <= 20, on the first line, which indicates the number of test cases.

Following this, there are *t* test cases on separate lines, each containing a single integer *Year*, 10 <= *Year* <= 10^6.

Output Format

For each test case, you should output, on a line by itself, the number of palindromic moments that occur during the year given in the test case.

Notes:

- We are interested in the total number of unique palindromic moments in a given year. If a moment is palindromic when using two or more different suffix/prefix combinations, this instance counts as a single palindromic moment. It should not be double counted.

  For example, `November 11, 2010 1:11:11 am` is palindromic when combining the `MMddyy` prefix and `Hmmss` suffix or the `MMddyy` prefix with the `HHmmss` suffix. Even though these combinations form two different palindromes, `11111011111` and `111110011111`, this should count as a single palindromic moment.

- You must account for the different number of days in each month and for leap years. However, you should ignore leap seconds.

Sample Input

```
2
2015
2016
```

Sample Output

```
5626
3761
```

Explanation

Test Case 1

There are 5626 palindromic moments during 2015. For example, the following list shows the 36 palindromic moments on January 1, 2015:

- 12:51:11 AM (using the `Mdyy` prefix and the `Hmmss` suffix): `111505111`
- 1:02:11 AM (using the `Mdyyyy` prefix and the `hmmss` suffix): `11201510211`
- 1:10:10 AM (using the `MMddyy` prefix and the `hmmss` suffix): `01011511010`
- 1:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111515111`
- 2:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111525111`
- 3:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111535111`
- 4:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111545111`
- 5:11:01 AM (using the `Mddyy` prefix and the `hmmss` suffix): `1011551101`
- 5:11:10 AM (using the `MMdyy` prefix and the `hmmss` suffix): `0111551110`
- 5:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111555111`
- 6:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111565111`
- 7:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111575111`
- 8:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111585111`
- 9:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `111595111`
- 10:21:01 AM (using the `Mddyyyy` prefix and the `hmmss` suffix): `1012015102101`
- 10:21:10 AM (using the `MMdyyyy` prefix and the `hmmss` suffix): `0112015102110`
- 11:51:11 AM (using the `Mdyy` prefix and the `hmmss` suffix): `1115115111`
- 1:02:11 PM (using the `Mdyyyy` prefix and the `hmmss` suffix): `11201510211`
- 1:10:10 PM (using the `MMddyy` prefix and the `hmmss` suffix): `01011511010`
- 1:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111515111`
- 2:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111525111`
- 3:11:01 PM (using the `Mddyy` prefix and the `Hmmss` suffix): `10115151101`
- 3:11:10 PM (using the `MMdyy` prefix and the `Hmmss` suffix): `01115151110`
- 3:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111535111`
- 4:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111545111`
- 5:11:01 PM (using the `Mddyy` prefix and the `hmmss` suffix): `1011551101`
- 5:11:10 PM (using the `MMdyy` prefix and the `hmmss` suffix): `0111551110`
- 5:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111555111`

- 6:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111565111`
- 7:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111575111`
- 8:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111585111`
- 9:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `111595111`
- 10:21:01 PM (using the `Mddyyyy` prefix and the `hmmss` suffix): `1012015102101`
- 10:21:10 PM (using the `MMdyyyy` prefix and the `hmmss` suffix): `0112015102110`
- 10:51:11 PM (using the `Mdyy` prefix and the `Hmmss` suffix): `1115225111`
- 11:51:11 PM (using the `Mdyy` prefix and the `hmmss` suffix): `1115115111`

In a similar way we may calculate the remaining palindromic moments for the rest of 2015, resulting in the reported overall sum of 5626 unique palindromic moments for the year.

Test Case 2

There are 3761 palindromic moments during 2016.

#9

Mr. Pippo's Pizza

Pizza's delicious. Pizza's nutritious.

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Mr. Pippo's Pizza

Problem Statement

Mr. Pippo wants to start a new pizza shop. Everything about his pizzas is unique — the recipe is unique, the taste is unique, and even the shape of pizzas is unique. Instead of having a round shape like every other pizza, Mr. Pippo makes his pizzas in polygon shapes. For example, he can make his pizzas in a triangular shape or in a pentagonal shape.

Before serving a pizza, Mr. Pippo cuts it into triangular pieces. However, there are different ways he can cut the pizza. For example, a pentagonal pizza can be cut in five different ways as shown in the following figure. Each day, Mr. Pippo chooses a particular shape which can only be cut in an odd number of ways. Note that all the five cuts in the figure happen to be rotationally symmetric, but each of them is considered distinct.

Figure: Different ways a pentagonal pizza can be cut

Your task in this problem is to determine the shape of the pizza. Given the number of ways the pizza can be cut, you have to determine how many sides the pizza has.

Further clarification regarding the ways a pizza can be cut is given below:

- A pizza can only be cut by connecting two vertices,
- Two cuts on the pizza cannot cross each other, and
- For an n-polygon there would be exactly (n-3) cuts which divide the pizza into (n-2) pieces.

Input Format

There will be up to 100 lines given where each line represents one test case. For each test case, the number of ways the pizza can be cut will be given. The number will be always odd and can be up to 308 digits long. The input is finished when end-of-file is reached.

Output Format

For each test case, print on a single line the number of sides the pizza has.

Sample Input

```
1
5
```

Sample Output

```
3
5
```

Explanation

For the first line of input, there is only one way to "cut" the pizza if the pizza is a triangle. This one way consists of no cuts at all.

For the second line of input, if there are five ways to cut the pizza, the pizza must be a pentagon, as shown in the figure in the problem description.

#10

Zoom In

Can you now see the "big picture"?

Zoom In

Problem Statement

Some years ago, we had terminals there were capable of supporting only ASCII characters. We would like your help to construct a program, which given an input string and specific printing rules, produces the same text in a bigger layout.

Input Format

On the first line of input is an integer n, $1 <= n <= 100$, representing how many columns each character will use when printed "zoomed-in".

The next line contains an integer m, $1 <= n <= 100$, representing how many rows each character will use when printed "zoomed-in". Note that n and m are not necessarily equal.

The third line contains an integer k, $3 <= k <= 95$, which indicates how many characters may need to be translated.

Following these first lines, are k descriptions of the "zoomed-in" characters, formatted as follows:

- On a line by itself, a single character, which has an ASCII value of between 32 and 126, inclusive
- m lines, each containing n characters, that give the "zoomed-in" representation of the character on the previous line

Following the descriptions of the zoomed in characters, is an integer number x, $1 <= x <= 500$.

Finally there are x lines, each containing a string of up to 2,000 characters, and ending with a new line. The characters in this string will be chosen from the set of k characters previously specified.

Notes:

- We don't know if k sets (i.e. the descriptions of the k "zoomed-in" characters) are given in a sorted or random order.
- The "zoomed-in" version of an empty string is m blank lines (i.e. lines with only a newline character).

Output Format

For each of the x strings, you should output the “zoomed-in” version. Each string should begin on a newline.

Note: You should perform only the transformation that is specified. You should not add any space between your printed letters that is not in the transformation.

Sample Input

```
4
4
3
H
H H
H%%H
H%%H
H H
i
 ()

 ||
```

```
 ||
!
 ||
 ||
 ||
 ()
1
Hi!
```

Sample Output

```
H  H () ||
H%%H    ||
H%%H || ||
H  H || ()
```

Explanation

For clarity, we will add dashes where the spaces would appear in the output in this explanation. According to the input, each character will use 4 rows and 4 columns, and there are 3 characters that may be translated.

A capital H ('H') should be translated as

```
H--H
H%%H
H%%H
H--H
```

A lower-case i ('i') should be translated as

```
-()-
----
-||-
-||-
```

An exclamation mark ('!') should be translated as:

```
-||-
-||-
-||-
-()-
```

We are then asked to print the "zoomed in" version of the string "Hi!". The output would be the following (with spaces where the dashes are located):

```
H--H-()--||-
H%%H-----||-
H%%H-||--||-
H--H-||--()-
```

#11

High Score

We have a new High Score.
Please enter your credentials...

High Score

Problem Statement

Find an input which will make the provided Java program give the highest output. An equivalent Python v.3 program is also provided. The programs are accessible via the "Programs" tab in the ribbon above with options: "Problem", "Submissions", "Leaderboard", "Discussions", and "Programs"

The best solution will get a full score, while others will receive an exponentially decaying score: losing 1% of the score for each unit decrease in the output.

Input Format

None.

Output Format

Your program should produce a legal input for the Java (or Python equivalent) program listed in the "Programs" tab.

The first line of of your solution's output should be a non-negative integer *N* representing the number of lines to follow.

The following *N* lines should each contain exactly 10 non-whitespace characters.

Sample Output

```
2
Abcde_-!@#
0#T234<>?,
```

Explanation

No input will be provided to your program. Your program should produce a fixed output according to the instructions provided above which will be then automatically provided as input to the given Java (or Python equivalent) program. For example, in this case, we may assume that you have written a program that outputs the 3 lines provided above as the *Sample Output*. This program outputs a value of 2 for *N*, and then two lines follow with 10 characters each.

When this output is provided as input to the Java program, it outputs a value equal to 998. As it turns out, this is not a very good input. The score for our solution would be:

$0.99^{MaxScore - 998}$

where *MaxScore* is the maximum possible score for this challenge.

MaxScore is large enough that this rounds to zero.

Note: The scores will be rounded to two decimal places.

Java Program

```
import java.util.Scanner;
public class Solution {
  public static boolean[] B = new boolean[50];
  public static char[] M = new char[256];
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    boolean[] I = new boolean[10];
    for (int i = 0; i < n; i++) {
      String s = sc.next();
      for (int j = 0; j < 10; j++) {
        I[j] = (s.charAt(j) & 1) != 0;
      }
      f(I);
    }
    System.out.println(1000*sum(M)-n);
  }
  public static int sum(char[] M) {
    int retVal = 0;
    for (int i = 0; i < M.length; i++)
      retVal += M[i];
    return retVal;
  }
  public static void f(boolean[] I) {
    boolean x1, x2, x3, x4, x5, x6, x7, x8, x9, y1, y2, y3, y4, y5, y6, y7, y8, y9;
    boolean c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c20, c21, c22, c23;
    boolean c30, c31, c32, c33, c34, c40, c41, c42, c43, c44, c45;
    boolean c51, c52, c53, c54, c55, c61, c62, c63, c64, c65, c73, c74, c75;
    boolean c84, c85, c95;
    B[49]=B[39];B[48]=B[38];B[47]=B[37];B[46]=B[36];B[45]=B[35];B[44]=B[34];
    B[43]=B[33];B[42]=B[32];B[41]=B[31];B[40]=B[30];B[39]=B[29];B[38]=B[28];
    B[37]=B[27];B[36]=B[26];B[35]=B[25];B[34]=B[24];B[33]=B[23];B[32]=B[22];
    B[31]=B[21];B[30]=B[20];B[29]=B[19];B[28]=B[18];B[27]=B[17];B[26]=B[16];
    B[25]=B[15];B[24]=B[14];B[23]=B[13];B[22]=B[12];B[21]=B[11];B[20]=B[10];
    B[19]=B[ 9];B[18]=B[ 8];B[17]=B[ 7];B[16]=B[ 6];B[15]=B[ 5];B[14]=B[ 4];
    B[13]=B[ 3];B[12]=B[ 2];B[11]=B[ 1];B[10]=B[ 0];B[ 9]=I[ 9];B[ 8]=I[ 8];
    B[ 7]=I[ 7];B[ 6]=I[ 6];B[ 5]=I[ 5];B[ 4]=I[ 4];B[ 3]=I[ 3];B[ 2]=I[ 2];
    B[ 1]=I[ 1];B[ 0]=I[ 0];
    x1= I[0]|I[1];x2= x1|I[2];x3= x2|I[3];x4= x3|I[4];x5= x4|I[5];x6= x5|I[6];
    x7= x6|I[7];x8= x7|I[8];x9= x8|I[9];
    y1=(!x9)|(I[0]&I[1]);y2=y1|(x1&I[2]);y3=y2|(x2&I[3]);y4=y3|(x3&I[4]);y5=y4|(x4&I[5]);
    y6=y5|(x5&I[6]);y7=y6|(x6&I[7]);y8=y7|(x7&I[8]);y9=y8|(x8&I[9]);
    c0=B[0]|B[10]|B[20]|B[30]|B[40];c1=B[1]|B[11]|B[21]|B[31]|B[41];
    c2=B[2]|B[12]|B[22]|B[32]|B[42];c3=B[3]|B[13]|B[23]|B[33]|B[43];
    c4=B[4]|B[14]|B[24]|B[34]|B[44];c5=B[5]|B[15]|B[25]|B[35]|B[45];
    c6=B[6]|B[16]|B[26]|B[36]|B[46];c7=B[7]|B[17]|B[27]|B[37]|B[47];
    c8=B[8]|B[18]|B[28]|B[38]|B[48];c9=B[9]|B[19]|B[29]|B[39]|B[49];
    c10=!(c0 | c1);
    c11=c0^c1;c12=c0&c1;
    c20=(c10&(!c2));c21=(c10&c2)|(c11&(!c2));c22=(c11&c2)|(c12&(!c2));c23=(c12&c2);
    c30=(c20&(!c3));c31=(c20&c3)|(c21&(!c3));c32=(c21&c3)|(c22&(!c3));
    c33=(c22&c3)|(c23&(!c3));c34=(c23&c3);c40=(c30&(!c4));c41=(c30&c4)|(c31&(!c4));
    c42=(c31&c4)|(c32&(!c4));c43=(c32&c4)|(c33&(!c4));c44=(c33&c4)|(c34&(!c4));
    c45=(c34&c4);c51=(c40&c5)|(c41&(!c5));c52=(c41&c5)|(c42&(!c5));
    c53=(c42&c5)|(c43&(!c5));c54=(c43&c5)|(c44&(!c5));c55=(c44&c5)|(c45&(!c5));
    c62=(c51&c6)|(c52&(!c6));c63=(c52&c6)|(c53&(!c6));c64=(c53&c6)|(c54&(!c6));
    c65=(c54&c6)|(c55&(!c6));c73=(c62&c7)|(c63&(!c7));c74=(c63&c7)|(c64&(!c7));
    c75=(c64&c7)|(c65&(!c7));c84=(c73&c8)|(c74&(!c8));c85=(c74&c8)|(c75&(!c8));
    c95=(c84&c9)|(c85&(!c9));
    boolean e=(!c95)|y9;
    boolean[] a= new boolean[10];
    a[0]=e|((((!c0)&(!c1)&(!c2)&(!c3)&(!c4))|(c0&c1&c2&c3&c4))^c0^c1^c2^c3^c4^(c3&
         (((c0^c8)&c1&c2&c4)^((((c0^c1)&c2&c5)^(c1&c4&c7))&c8))));
    a[1]=e|((((!c0)&(!c1)&(c2)&(!c5)&(c6))|(c0&c1&((!c2)&(!c6))&c5))^c0^c1^c2^c5^c6^(c4&
         ((c0&c1&((c2&c3)^(c5&c6)))^(((c1&c7)^(c6&c9))&c3&c8))));
    a[2]=e|((((!c0)&(!c1)&(c3)&(!c5)&(!c7))|(c0&c1&(!c3)&c5&c7))^c0^c1^c3^c5^c7^(c0&c1&c2&
         (c3^c4)&c5)^((c3^c4)&c5&c7&c8&c9));
    a[3]=e|((c3&c5)^(c3&c6)^(c3&c8)^(c3&c9)^(c5&c6)^(c5&c8)^(c5&c9)^(c6&c8)^(c6&c9)^
```

```
        (c8|c9)^c3^c5^c6^c8^c9^(c0&c1&c3&c6&c9));
    a[4]=e|((c2&c5)^(c2&c7)^(c2&c8)^(c2&c9)^(c5&c7)^(c5&c8)^(c5&c9)^(c7&c8)^(c7|c9)^
        (c8&c9)^c2^c5^c7^(((c0&c5&c6)^(c1&c3&c4))&c7&c8));
    a[5]=e|((c0&c1)^c0^c2^c4^c6^c7^(c0&c1&c2&c3&c4)^(((c0&((c3&c5)^(c2&c4)))^
        (c1&c4&c6))&c7&c8)^(c3&c4&c6&((c2&c9)^(c5&c7))));
    a[6]=e|(c0^c1^c3^c4^c7^(c0&c1&c2&c4&c9)^(c0&((c1&c4)^(c3&c8))&c5&c7)^
        ((((((c0^c1)&c5)^(c0&c4))&c2)^(c1&(c2^c7)&c4))&c6&c8));
    a[7]=e|(c2^c3^c4^(c0&((c2&c3)^((c2^c3)&c7))&c4&c8)^((((c0^c1)&c3&c5)^(((c0^c1)&
        (c4^c5))&c6))&c7&c8));
    int val = 0;
    for (int i = 7; i >= 0; i--) {
      val = val << 1;
      if (a[i]) val = val | 1;
    }
    M[val]=1;
  }
}
```

Python3 Program

```
I=10*[]
B=50*[0]
M=256*[0]
def f(I):
  B[49]=B[39];B[48]=B[38];B[47]=B[37];B[46]=B[36];B[45]=B[35];B[44]=B[34];B[43]=B[33];
  B[42]=B[32];B[41]=B[31];B[40]=B[30];B[39]=B[29];B[38]=B[28];B[37]=B[27];B[36]=B[26];
  B[35]=B[25];B[34]=B[24];B[33]=B[23];B[32]=B[22];B[31]=B[21];B[30]=B[20];B[29]=B[19];
  B[28]=B[18];B[27]=B[17];B[26]=B[16];B[25]=B[15];B[24]=B[14];B[23]=B[13];B[22]=B[12];
  B[21]=B[11];B[20]=B[10];B[19]=B[ 9];B[18]=B[ 8];B[17]=B[ 7];B[16]=B[ 6];B[15]=B[ 5];
  B[14]=B[ 4];B[13]=B[ 3];B[12]=B[ 2];B[11]=B[ 1];B[10]=B[ 0];B[ 9]=I[ 9];B[ 8]=I[ 8];
  B[ 7]=I[ 7];B[ 6]=I[ 6];B[ 5]=I[ 5];B[ 4]=I[ 4];B[ 3]=I[ 3];B[ 2]=I[ 2];B[ 1]=I[ 1];
  B[ 0]=I[ 0];
  x1= I[0]|I[1];x2= x1|I[2];x3= x2|I[3];x4= x3|I[4];x5= x4|I[5];x6= x5|I[6];x7= x6|I[7];
  x8= x7|I[8];x9= x8|I[9];
  y1=(not x9)|(I[0]&I[1]);y2=y1|(x1&I[2]);y3=y2|(x2&I[3]);y4=y3|(x3&I[4]);
  y5=y4|(x4&I[5]);y6=y5|(x5&I[6]);y7=y6|(x6&I[7]);y8=y7|(x7&I[8]);y9=y8|(x8&I[9]);
  c0=B[0]|B[10]|B[20]|B[30]|B[40];c1=B[1]|B[11]|B[21]|B[31]|B[41];
  c2=B[2]|B[12]|B[22]|B[32]|B[42];c3=B[3]|B[13]|B[23]|B[33]|B[43];
  c4=B[4]|B[14]|B[24]|B[34]|B[44];c5=B[5]|B[15]|B[25]|B[35]|B[45];
  c6=B[6]|B[16]|B[26]|B[36]|B[46];c7=B[7]|B[17]|B[27]|B[37]|B[47];
  c8=B[8]|B[18]|B[28]|B[38]|B[48];c9=B[9]|B[19]|B[29]|B[39]|B[49];
  c10=not (c0 | c1);c11=c0^c1;c12=c0&c1;
  c20=(c10&(not c2));c21=(c10&c2)|(c11&(not c2));c22=(c11&c2)|(c12&(not c2));
  c23=(c12&c2);c30=(c20&(not c3));c31=(c20&c3)|(c21&(not c3));
  c32=(c21&c3)|(c22&(not c3));c33=(c22&c3)|(c23&(not c3));c34=(c23&c3);
  c40=(c30&(not c4));c41=(c30&c4)|(c31&(not c4));c42=(c31&c4)|(c32&(not c4));
  c43=(c32&c4)|(c33&(not c4));c44=(c33&c4)|(c34&(not c4));c45=(c34&c4);
  c51=(c40&c5)|(c41&(not c5));c52=(c41&c5)|(c42&(not c5));c53=(c42&c5)|(c43&(not c5));
  c54=(c43&c5)|(c44&(not c5));c55=(c44&c5)|(c45&(not c5));c62=(c51&c6)|(c52&(not c6));
  c63=(c52&c6)|(c53&(not c6));c64=(c53&c6)|(c54&(not c6));c65=(c54&c6)|(c55&(not c6));
  c73=(c62&c7)|(c63&(not c7));c74=(c63&c7)|(c64&(not c7));c75=(c64&c7)|(c65&(not c7));
  c84=(c73&c8)|(c74&(not c8));c85=(c74&c8)|(c75&(not c8));c95=(c84&c9)|(c85&(not c9));
  e=(not c95)|y9;
  a=10*[0]
  a[0]=e|((((not c0)&(not c1)&(not c2)&(not c3)&(not c4))|(c0&c1&c2&c3&c4))^c0^c1^c2^ \
       c3^c4^(c3&(((c0^c8)&c1&c2&c4)^((((c0^c1)&c2&c5)^(c1&c4&c7))&c8))));
  a[1]=e|((((not c0)&(not c1)&(    c2)&(not c5)&(    c6))|(c0&c1&((not c2)& \
       (not c6))&c5))^c0^c1^c2^c5^c6^(c4&((c0&c1&((c2&c3)^(c5&c6)))^
       (((c1&c7)^(c6&c9))&c3&c8))));
  a[2]=e|((((not c0)&(not c1)&(    c3)&(not c5)&(not c7))|(c0&c1&(not c3)&c5&c7))^c0^ \
       c1^c3^c5^c7^(c0&c1&c2&(c3^c4)&c5)^((c3^c4)&c5&c7&c8&c9));
  a[3]=e|((c3&c5)^(c3&c6)^(c3&c8)^(c3&c9)^(c5&c6)^(c5&c8)^(c5&c9)^(c6&c8)^(c6&c9)^ \
       (c8|c9)^c3^c5^c6^c8^c9^(c0&c1&c3&c6&c9));
  a[4]=e|((c2&c5)^(c2&c7)^(c2&c8)^(c2&c9)^(c5&c7)^(c5&c8)^(c5&c9)^(c7&c8)^(c7|c9)^ \
       (c8&c9)^c2^c5^c7^(((c0&c5&c6)^(c1&c3&c4))&c7&c8));
  a[5]=e|((c0&c1)^c0^c2^c4^c6^c7^(c0&c1&c2&c3&c4)^(((c0&((c3&c5)^(c2&c4)))^ \
       (c1&c4&c6))&c7&c8)^(c3&c4&c6&((c2&c9)^(c5&c7))));
```

```
  a[6]=e|(c0^c1^c3^c4^c7^(c0&c1&c2&c4&c9)^(c0&((c1&c4)^(c3&c8))&c5&c7)^ \
       ((((((c0^c1)&c5)^(c0&c4))&c2)^(c1&(c2^c7)&c4))&c6&c8));
  a[7]=e|(c2^c3^c4^(c0&((c2&c3)^((c2^c3)&c7))&c4&c8)^((((c0^c1)&c3&c5)^(((c0^c1)& \
       (c4^c5))&c6))&c7&c8));
  M[sum([a[i] << i for i in range(8)])]=1
output=int(input())
for i in range(output):
  f([ord(c)&1 for c in input()])
output = 1000*sum(M)-output
print(output)
```

#12

Threesome Poker

Or is it Three Sum Poker?

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Threesome Poker

Problem Statement

Help IBM puzzlemaster to test answers to May's 2015 challenge:

> Three people are playing the following betting game.
>
> Every five minutes, a turn takes place in which a random player rests and the other two bet against one another with all of their money. The player with the smaller amount of money always wins, doubling his money by taking it from the loser.
>
> For example, if the initial amounts of money are 1, 4, and 6, then the result of the first turn can be either 2,3,6 (1 wins against 4); 1,8,2 (4 wins against 6); or 2,4,5 (1 wins against 6). If two players with the same amount of money play against one another, one of the two players would win, and the game will immediately end for all three players after that turn.
>
> The task for the challenge is to find initial amounts of money for the three players, where none of the three has more than 255, and in such a way that the game cannot end in less than one hour.
>
> In the example above (1,4,6), there is no way to end the game in less than 15 minutes. One possible sequence of moves is:
>
> (1,4,6)
> (1,8,2), after player 2 plays player 3 in turn 1
> (2,7,2), after player 1 plays player 2 in turn 2
> (4,7,0), after player 1 plays player 2 in turn 3, and we arbitrarily choose player 2 as the winner
>
> Note: The first turn is played after five minutes, so the game above, with its 3 turns, takes 15 minutes. Furthermore, the maximum number of turns that can be completed in one hour is 11.

Input Format

Three space-separated, positive integers representing a potential solution to the challenge.

Output Format

The program should output either the word "Ok" if the answer is correct (i.e. the game needs an hour or longer to end), or in case there exists a series of turns in which a player wins the game in less than an hour, then the program should output at most 12 lines showing the amounts of money that each player has. In particular, in the latter scenario, the first line of the output should be identical to the given input, and then each subsequent line should represent a possible state of the game after a turn is played, with the previous line considered as a starting state. Since by definition in this scenario there will be a winner, one of the amounts in the last line of the output should be zero.

If there are several ways for the game to end in less than an hour, find the shortest one, i.e. the one with the fewest turns. If there are several shortest ones, choose the one in which the winning player and losing player in each round is first in lexicographical order. Please refer to the explanation of the sample input and output for more information about this tie breaking procedure.

Sample Input

```
3 27 8
```

Sample Output

```
3 27 8
6 24 8
6 16 16
6 32 0
```

Explanation

The first line of the output is identical to the given input. Then, in the first turn of the game (second line of the output), player 1 beats player 2. In the second turn (third line of the output), player 3 beats player 2. In the final round (final line of the output) player 2 beats player 3.

This game-play can be summarized in a single line comprised of space separated pairs that describe the winner and the loser of each turn in the format: ([winner],[loser]). For the example above, this string summary would be: "(1,2) (3,2) (2,3)" since at first player 1 beat player 2, then player 3 beat player 2 and finally the game ended with player 2 winning over player 3.

Of course, other game-plays could also result in the same result (i.e. have a player winning the game in less than an hour), some with more turns than the one described above, but some as equally as short (in term of number of turns). For example, such another equally short series of turns is the following:

```
3 27 8
3 19 16
6 16 16
6 0 32
```

However, this game-play would be summarized by the following string: "(3,2) (1,2) (3,2)" since at first player 3 beats player 2, then player 1 beats player 2 and finally the game ended with player 3 winning over player 2.

Note though that the last solution has a string representation that comes later in lexicographical order compared to our first solution (i.e. solution "(1,2) (3,2) (2,3)" comes before "(3,2) (1,2) (3,2)" in lexicographical order).

Similarly, another possible series of turns would be:

```
3 27 8
3 19 16
6 16 16
6 0 32
```

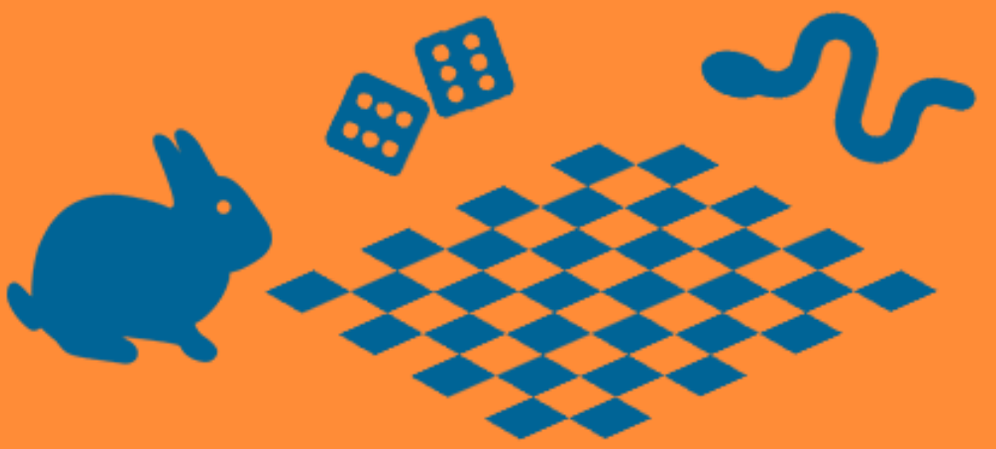
This series would be represented by the string: "(3,2) (1,2) (2,3)", which also comes after "(1,2) (3,2) (2,3)" in lexicographical order.

Indeed if you listed all possible games that end in three turns, "(1,2) (3,2) (2,3)" would be lexicographically smaller than all other string representations of games. Thus the expected output should be:

```
3 27 8
6 24 8
6 16 16
6 32 0
```

Note: There are two additional sample test cases. You can view these cases after clicking on the "Run" button.

#13



Snakes & Bunnies

Bunnies strike back!

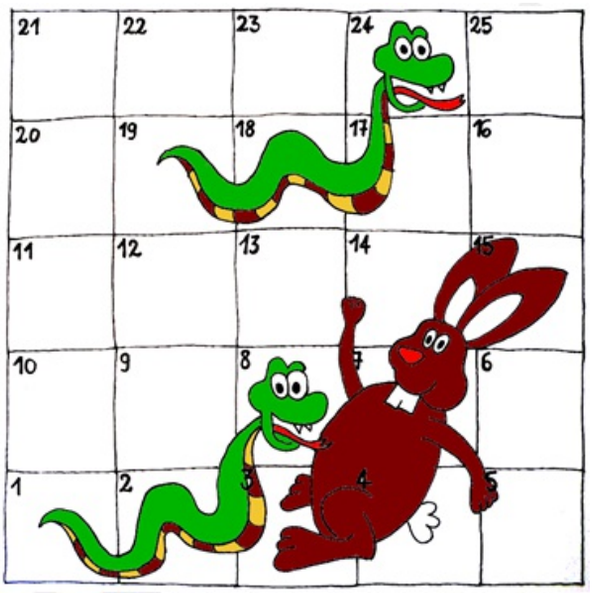
IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Snakes & Bunnies

Problem Statement

A group of engineers are playing the Snakes and Bunnies board game. In this game, each player has one game piece, which moves according to dice rolls. The game-board is composed by a grid of $N \times N$ numbered squares, which are ordered row-wise from the bottom-left to the top-right (N is always odd). Some “bunnies” and “snakes” are depicted on the board, each connecting two squares. An example of a 5x5 board is shown in the following figure.



Each player starts off the board, next to the starting square. The gameplay is divided in rounds and players play in sequence, which is the same every round. Rules are simple:

1. Two standard 6-faced die are rolled by the player and his/her game piece moves forward on the board, following the square's numbers. The piece is advanced by a number of squares equal to the sum of the die.

2. If the dice roll is a double, then the player has an additional turn just after the current one. Note that the additional turn begins after applying the additional rules below. The additional turn follows the same rules of standard turns, except that only one die can be rolled.

3. If a square is already taken by another player’s token, then the current player’s token moves forward to the next square not occupied by a token.

4. If the final position of a player’s token is a square with the head of a snake, then it must be moved backwards to the square corresponding to the snake’s tail. Similarly, if the token ends on a square with bunny’s feet, it goes to the top of the bunny’s ears (both ears of the rabbit will always point to the same cell). No square has two or more snake's heads/tails or bunny's ears/feet on it; there is at most one special drawing for each cell. Moreover, the last square is always free of drawings.

5. Game ends when a player arrives at the last square or when a player can move over the last square (for example if the player is on the second-to-last square and rolls 3+4). In this latter case, the player stops on the last square and wins.

Note that infinite loops can happen while a player moves: this is the EVIL CYCLE case! When it happens, the game ends and the player in the evil cycle wins the game.

For example, consider a board in which squares 10 and 11 are already occupied, and there is a slide from

square 12 back to square 10. If a player lands on square 10, they would advance to square 11, since square 10 is already occupied. Square 11 is also occupied, so they would advance to square 12. Here they take the slide back to square 10, and then repeat the moves. They would continue to move between squares 10, 11, and 12 forever!

Your task is to find the final position of every player's token, given as input the number of players, the game-board configuration, and the sequence of dice rolls.

Notes:

- The given sequence of dice rolls may not always lead a game to an end. There will be no extra dice rolls after a game has ended. There will always be sufficient dice roles for a player to complete their turn.
- If a player lands on the tail of snake or the ears of a bunny, the player does not make any special moves.

Input Format

The first line of the input contains the integer N ($1 < N < 100$ and N is odd), which is the dimension of the game-board.

The following N lines contain the game-board configuration: each line contains N characters, and each character represents a square of the board. The character '-' represents a normal square, i.e. one with no snake heads/tails nor bunny feet/ears depicted on it; digits (0-9) represent bunnies and letters (a-z) represent snakes. There can be at most 10 bunnies and 26 snakes, and each represented by an appropriate pair of digits or letters. Given a pair of identical letters representing a snake in two numbered squares, the head of the snake are located in the square with the higher number, and the tail is located in the square with the lower number. Given a pair of identical digits representing a bunny in two numbered squares, the feet of the bunny is located in the square with the lower number, and the ears are located in the square with the higher number.

So, for example, the game-board of the figure above could be represented as follows:

```
---a-
-a---
----1
--b--
b-1--
```

After the game board representation, the following line contains an integer M ($2 <= M <= 10$), which is the number of players.

Then, the sequence of dice throws follow, each dice result is represented by a single line containing one integer between 1 and 6, inclusive.

Output Format

The output is a single line containing M integers separated by a blank space. The first integer is the final position on the game-board of the first player (i.e. the one who rolled the dice first), the second integer is the final position of the second player, etc.

In case of evil cycle, the output is a single line containing the string “PLAYER x WINS BY EVIL CYCLE!”, where x is the player number (1 to M).

Sample Input

```
5
---a-
-a---
----1
--b--
b-1--
2
6
2
2
1
3
2
1
2
3
4
1
1
5
```

Sample Output

```
13 25
```

Explanation

The board in this test case is the board in the figure below.

21	22	23	24 **SNAKE HEAD**	25
20	19 **SNAKE TAIL**	18	17	16
11	12	13	14	15 **BUNNY EARS**
10	9	8 **SNAKE HEAD**	7	6
1 **SNAKE TAIL**	2	3 **BUNNY FEET**	4	5

Note that in the board above there is one bunny, represented by the number '1'. It's feet are at square 3 and its ears are at square 15. There are two snakes. The snake indicated by the letter 'b' has a head at square 8 and a tail at square 1. The snake indicated by the letter 'a' has a head at square 24 and a tail at square 19.

Next we are told that there are 2 players.

First, player 1 rolls a 6 and a 2, advancing to square 8, at the head of the snake. Here, he moves back to the snake's tail, ending at square 1.

Next, player 2 rolls a 2 and a 1, advancing to square 3. Since 3 is at the feet of a bunny, she advances to square 15.

Next, player 1 rolls a 3 and a 2, advancing to square 6.

Next, player 2 rolls a 1 and a 2, advancing to square 18.

Next, player 1 rolls a 3 and a 4, advancing to square 13.

Next, player 2 rolls double 1's, advancing to square 20. Since the player rolled doubles, player 2 rolls a single die and gets a 5. She then advances to square 25.

Since player 2 has reached the final square, the game is over.

Note: If you click on the "Run Code" button, you will be able to see an additional sample test case with an example of an EVIL CYCLE.

#14

IEEE State of Mind

2B OR NOT 2B that is the question.

IEEE State of Mind

Problem Statement

Finite State Machines (FSMs) can be used to model systems whose output depends on the current state and its inputs. The states can represent many different systems like the on/off state of traffic lights at an intersection, the valve state of thrusters on a spacecraft, the click/key combinations entered by a user for a keyboard shortcut, or the send/receive steps in a communication protocol.

An FSM is defined by a set of states, inputs, and outputs. The states represent a known configuration of the system at a specific time. Transitioning to a new state depends on the current state and the inputs provided. This will in turn also produce an associated output for the system.

One of the uses of FSMs is to design and analyze logic circuits with memories like flip-flops. Your task is to write a program to visualize the timing diagram relating the inputs and outputs while printing the current state number to help analyze the circuit.

For the purpose of the program, you can assume that there are N states with M inputs. States are labeled with numbered subscripts in the range S_0 to S_{N-1}. The inputs are each labeled with a single uppercase character in the range 'A' to 'J', inclusive. You may assume that no letter or state number is skipped. Each state has P uniquely defined transitions to other states (the transitions are unique in the sense that no set of inputs will trigger more than 1 transition). A transition is only taken when the input conditions are satisfied. Transitions are evaluated at the beginning of each of the T discrete time steps.

Input Format

Input begins with a line containing N and M, where $1 \leq N < 100$ and $1 < M < 10$.

Then there are N lines, each describing the output the system is in a state and the transitions from this state. The first line contains a description of state S_0, the second line describes S_1, etc. These lines all have the following format:

$StateOutput\ P\ VariableExpression_1/Dest_1\ VariableExpression_2/Dest_2\ \ldots\ VariableExpression_P/Dest_P$

where

- $StateOutput$ is either a 0 or 1, representing the value that is output when the system is in this state
- P is the number of transitions from this state
- $VariableExpression_i$ is a comma-delimited list of $Variable=Value$ tokens. For a transition i to be activated, all of the input variables given in the $VariableExpression_i$ must have the values listed in this expression
- $Dest_i$, $1 <= i <= P$, is a destination state number, in the range [0..N), for transition i

Following the description of the states, is a line containing two integers, T and I, where T gives the number of timesteps ($1 <= T < 1000$), and I gives the number of the initial state ($0 <= I < N$).

The input ends with $M * T$ lines that provide the inputs for all M variables in each of the T timesteps. The first M values provide inputs for the M variables during the first timestep, the second M values provide inputs for the M variables during the second timestep, and so on.

The transitions should be interpreted as follows. Assume that we are in a state S_j. Assume further that there are four inputs, and the current input provided for the current timestep is (1,0,1,0). We map these inputs to

the variables, starting with letter 'A', so A = 1, B = 0, C = 1, and D = 0. We would then look for transitions defined on the line corresponding to state S_j. Since transitions must be unique, only one of the following *VariableExpressions* could appear in the list of expressions: `A=1`, `B=0`, `C=1`, `D=0`, `A=1,B=0`, `A=1,C=1`, `A=1,D=0`, `A=1,B=0,C=1`, etc. If such a transition is the i^{th} transition specified, the new state would be $Dest_i$. If there are no matches, the system remains in the current state.

Output Format

The output consists of a waveform. The waveform is represented by underscores, '_', for the number 0, and asterisks, '*', for the number 1.

A maximum of 16 ticks are printed together, where each time tick is 3 characters wide. The line numbers are labeled with the signal names as shown in the example.

As shown in the example below, on the first line of the grouping is the text `Tick #X` where `X` is replaced by the number of the first tick in the grouping.

Next come waveforms for each of the variables, in alphabetical order, one per line. On these lines the variable name is output, followed by five spaces, and then the waveform.

Then the waveform of the system output is displayed. This line begins with the word `OUT`, followed by 3 spaces, and then the waveform.

Finally, on the last line the numeric value of the system output is displayed. This line begins with the word `STATE`. The numeric values of system output should always be aligned with the third column of the respective tick.

If not all ticks have been displayed, then a blank line should be output, followed by the next group of ticks in the same format.

Sample Input

```
2 3
1 2 A=1,B=1,C=1/1 A=1,B=0/0
0 1 A=0/0
20 0
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
0 0 0
0 0 1
0 1 0
0 1 1
```

Sample Output

```
Tick #1
A   __________**********__________**********
B   _____******_____******_____******_____******
C   ___***___***___***___***___***___***___***___***
```

```
OUT   ********************__********************__
STATE  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

Tick #17
A    ________________
B    ________********
C    __****__****
OUT  ****************
STATE  0 0 0 0
```

Explanation

The sample input executes a parity check FSM. There are N=2 states with M=3 inputs. Since there are 3 inputs, these inputs are labeled `A`, `B`, and `C`.

State 0, S_0, outputs the value `1` (logic high). There are two transitions originating from this state. The first transition goes to S_1 when input `A`=1, `B`=1, and `C`=1. The second transition goes to S_0 when input `A`=1 and `B`=0 (that's a self-loop to the same state). `C` can take any value in the second transition.

S_1 outputs the value `0` (logic low). There is only one transition originating from this state defined in the input. This transition goes to S_0 when `A`=0.

Note that although not defined, there are implicit transitions to satisfy the other cases that are not explicitly described in the input. The implicit transitions are all self-loops to the same state.

Prior to tick t=1, we are in the initial state, S_0. The input is set to `A`=0, `B`=0, and `C`=0, therefore we take the implicit self-loop transition and stay in S_0 and output the value associated with the state, i.e. `1`.

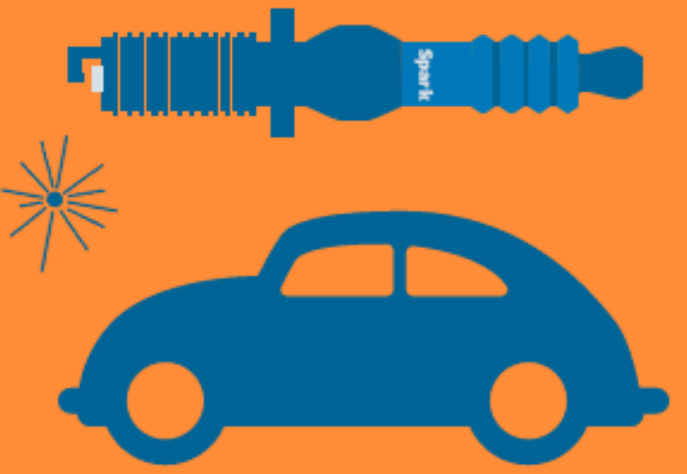
When we start ticks t=2 through t=7, we are still in S_0. In each of these cases, we take the implicit loop to the same state and nothing changes in the output.

In tick t=8, the inputs, `A`=1, `B`=1, and `C`=1, trigger the transition to state S_1. This changes the output of our logic circuit to a `0`.

In tick t=9, the input, `A`=0, trigger the transition to state S_0. This changes the output of our logic circuit to a `1`.

The same pattern of inputs is repeated a few more times. Tick 17 begins on a new line because we limit the output to 16 ticks per line.

#15



Car Spark

Help launch the Next Big Thing

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Car Spark

Problem Statement

John is a computer programmer and is highly known for his achievements in his field. In addition to being a passionate software professional, he is also passionate about motorcars and motorbikes.

So, after ending his successful and lengthy software career, he decides to take up his passion. He starts an organization by the name "Car Spark" (CS). CS is an organization from which you can rent luxury cars of your choice on an hourly rental basis.

CS would like to accept bookings for the weekend in advance, and then decide which bookings to process based on the profits that would be earned. When placing an order, customers quote the amount that they are willing to pay for that vehicle during that particular timespan. Since a car can only be given to one customer during a particular time period, CS must be careful about which bookings to process.

Initially CS has only one vehicle available for rent. To be the first hire for CS, you must develop a program to maximize revenue on bookings for this vehicle.

Input Format

Input begins with a single integer T, 1 <= T <= 100, which denotes number of test cases.

Each test case begins with a single integer N, 1 <= N <= 2000, which is the number of bookings John received.

The remainder of the test case consists of N lines containing three integers B_s, B_e, and A_i each separated by a space, where B_s is the booking start time, B_e is the booking end time, and A_i is the amount that the customer is willing to spend for the entire booking. Note that 0 <= B_s < B_e <= 48 and 1 <= A_i <= 100000.

Note: The car may only be rented during the weekend, meaning from 12:00 AM on Saturday to 12:00 AM on Monday. Since the two days in the weekend have 48 hours, 12 noon on a Sunday would be the (24+12) 36th hour. Similarly, if the booking start time is 10:00 PM on Saturday and the booking end time is 12:00 AM on Sunday, then B_s would be 22 and B_e would be 24.

Output Format

You are to output a single line for each test case, giving the maximum revenue John can make from the orders he received.

Sample Input

```
2
4
1 2 100
2 3 200
3 4 1600
1 3 2100
3
1 10 2000
2 5 100
6 9 400
```

Sample Output

```
3700
2000
```

Explanation

For the first test case, for the time slot 1-3 maximum revenue John can make is 2100 (Max(100+200, 2100)) and for slot 3-4 he can make 1600. The maximum total revenue is 3700 (2100+1600).

Similarly for second test case, the maximum revenue he can generate is 2000.

#16

Dictionary Strings

If a word in the dictionary were misspelled, how would we know? :)

Dictionary Strings

Problem Statement

Gopal is preparing for a competitive exam and he has to prepare many topics for it. To remember the concepts better he identified a set of words from each topic. He prepared dictionaries for each of these topics with the set of identified words so that he can refer to them easily.

While recollecting the topics Gopal sometimes could not remember to which dictionary a certain word belongs. After all the hard work, Gopal didn't want to lose marks due to this confusion. So he requested his friend, Govind, to help him identify a way to check if a word belongs to a dictionary.

Govind, being a very good friend of Gopal, wants to help him do better in the exam. So, after some thought, he finally came up with a solution.

For each dictionary, a string is chosen from which all the words can be made by selecting a subset of the characters from the string and rearranging them. (It is not necessary that the characters are consecutive and/or in the same order as in the string). They called this string a `Dictionary String`. When confused about to which dictionary a word belongs, Gopal can check if the word can be extracted from the `Dictionary String` for that dictionary.

To qualify as a `Dictionary String`, all the letters needed to explicitly form each word of the dictionary must be present in the string. You cannot reuse letters. Thus, the string `aab` is not a Dictionary String for a dictionary containing the word `aaa` since this word needs 3 `a`'s whereas the candidate Dictionary String has only two `a`'s.

To help Gopal memorize the Dictionary Strings better, Govind inserted extra characters in some of the Dictionary Strings that appeared harder to memorize. To distinguish those strings from others he calls a string without any extra characters, a `Perfect Dictionary String`.

Govind would like your help in verifying his program. For a set of words in a dictionary, you should indicate whether a string is a perfect dictionary string and/or a dictionary string. If a word is not a dictionary string, he would like you to tell him the minimum number of characters needed to convert the string to a dictionary string.

Notes:

Some of the test cases are very large, and may require you to speed up input handling in some languages.

In C++, for example, you can include the following line as the first line in your main function to speed up the reading from input:

```
std::ios_base::sync_with_stdio (false);
```

And in Java, you can use a BufferedReader to greatly speed up reading from input, e.g.:

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
// Read next line of input which contains an integer:
int T = Integer.valueOf(reader.readLine());
```

Input Format

Input begins with a single integer T, $1 <= T <= 100$, which denotes number of test cases.

Each test case begins with a line, which contains 2 space-separated integers D and S. D represents the number of words in a dictionary, and S represents the number of potential dictionary strings to be checked. Note that $1 <= D, S <= 100$.

Next follows D lines, each containing a word in the dictionary.

The remaining S lines in the test case each contain a potential dictionary string.

Notes: The words in the dictionary and the potential dictionary strings will consist of only lower-case letters. The lengths of these strings are greater than or equal to one character and less than or equal to 40,000 characters.

Output Format

For each of the S potential dictionary strings, you should output a line with two values separated by a space in the following format:

A_1 A_2

Where

- A_1 is either `Yes` or `No` denoting if a string is a Dictionary String or not.
- If A_1 is `No`, then A_2 is the minimum number of characters needed to make the string a Dictionary String. If A_1 is `Yes`, then A_2 is `Yes` if the string is a Perfect Dictionary String, and `No` otherwise.

Sample Input

```
1
5 3
ant
top
open
apple
lean
anteplop
antelope
penleantopan
```

Sample Output

```
Yes Yes
No 1
Yes No
```

Explanation

For the sample input, there is only one test case with 5 words in it and 3 strings to be checked.

`anteplop`: contains all the words from the dictionary and no extra characters. So it is both a Dictionary String and a Perfect Dictionary String. Hence, the output is `Yes Yes`.

`antelope`: the words “apple” cannot be made from this string. So it is not a Dictionary String and is missing 1 character (‘p’) to become a dictionary string. Hence the output `No 1`.

`penleantopan`: all the words of the dictionary can be made from this string but it also contains extra characters that are not required to build the words of the dictionary. So it is a Dictionary String but not a Perfect Dictionary String.

#17

Light Gremlins

Now we know who has been leaving the lights on.

Light Gremlins

Problem Statement

There are a group of gremlins that live in a long hallway in which there are a series of light switches. At the beginning of each night, all of the light switches are off. Then, one at a time, each gremlin does the following:

- The gremlin chooses a prime number *p*, that has not been chosen by any previous gremlin that night.
- The gremlin runs down the hallway flipping every p^{th} switch.

The owner of the hallway, who is very concerned about his electricity bill, has asked you to determine how many switches are on at the end of the night.

Note: no two gremlins will choose the same prime number.

Consider the following example where the hallway has 21 switches and there are three gremlins. At the beginning of the night, all switches are off, as shown in the figure below.

Switch:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
State:	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off

The first gremlin chooses the prime number 7, and flips the 7^{th}, 14^{th}, and 21^{st} switch. Now the configuration is:

Switch:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
State:	Off	Off	Off	Off	Off	Off	**On**	Off	Off	Off	Off	Off	Off	**On**	Off	Off	Off	Off	Off	Off	**On**

The second gremlin chooses the prime number 13, and flips just the 13^{th} switch, because there is no 26^{th} switch. Now the configuration is:

Switch:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
State:	Off	Off	Off	Off	Off	Off	**On**	Off	Off	Off	Off	Off	**On**	**On**	Off	Off	Off	Off	Off	Off	**On**

The last gremlin chooses the prime number 3. It flips the 3^{rd}, 6^{th}, 9^{th}, 12^{th}, 15^{th}, 18^{th}, and 21^{st} switch. Note that when he flips the 21^{st} switch, it is turned back off. The final configuration is:

Switch:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
State:	Off	Off	**On**	Off	Off	**On**	**On**	Off	**On**	Off	Off	**On**	**On**	**On**	**On**	Off	Off	**On**	Off	Off	Off

For this example, you would report that there are 9 lights on at the end of the night.

Input Format

The input begins with an integer *t*, 1 <= *t* <= 20, on a line by itself.

Then follow *t* lines, each describing a test case that you must evaluate. The test cases have the following format:

```
[switch] [n] [prime_1] [prime_2] ... [prime_n]
```

Where

- [switch] is the number of switches in the hallway, 1 <= [switch] <= 10^{18}

- [n] is the number of gremlins who live in the hallway, 1 <= [n] <= 24
- The prime number chosen by the i^{th} gremlin is given by [prime_i]. All primes are greater than or equal to 2 and less than 10^4.

Output Format

For each test case, you should output a single integer that indicates how many switches are on at the end of the night.

Sample Input

```
3
21 3 7 13 3
20 1 31
30 3 2 3 5
```

Sample Output

```
9
0
15
```

Explanation

The first test case corresponds to the example given in the Problem Definition, which as described above results in 9 "on" switches at the end of the night.

In the second test case, there is a single gremlin, who chooses the prime 31. The hallway consists of only 20 switches, so there is no 31^{st} switch. Thus, no switches are turned on.

The last test case consists of a hallway of length 30, and three gremlins. The action of the gremlins is as follows:

- The first gremlin flips switches {2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30}. All of these switches were previously off, so they are now on.
- The second gremlin flips switches {3, 6, 9, 12, 15, 18, 21, 24, 27, 30}. Of these, {6, 12, 18, 24, 30} were previously on, so they are now off. This results in the following switches being on: {2, 3, 4, 8, 9, 10, 14, 15, 16, 20, 21, 22, 26, 27, 28}.
- The third gremlin flips switches {5, 10, 15, 20, 25, 30}. Of these, {10, 15, 20} were previously on, so they are now off. This results in the following switches being on: {2, 3, 4, 5, 8, 9, 14, 16, 21, 22, 25, 26, 27, 28, 30}.

Thus, there are 15 switches on at the end of the night.

#18

Chocolate

Chocolate, how could I eat you? Let me count the ways.

Chocolate 2

Problem Statement

Let us consider a rectangular bar of chocolate containing n times d chunks of chocolate. Each chunk can be uniquely identified by a pair of integers (i_1, i_2), where each coordinate i_1 indentifies the row of the chunk and i_2 identifies the column. i_1 can take values between 1 and n, inclusive, and i_2 can take values between 1 and d, inclusive. The chunk at the top left corner of the bar is identified by (1,1).

One wants to eat the whole bar using a very elaborate method. The principle is that when the chunk with coordinates (i_1,i_2) is selected to be eaten, all remaining chunks with coordinates (j_1, j_2) such that $j_1 >= i_1$ and $j_2 >= i_2$ are eaten at the same time. The question we ask is:

Given n and d, how many different ways of eating the whole bar are there?

Note: The timeouts have been increased by approximately 50% for this problem.

Input Format

The first line of input contains the integer n. The second line of input contains the integer d. Note that 1 <= n, d <= 100. Furthermore, n and d are chosen such that the maximum number of ways of eating the whole bar will never exceed 10^{138}.

Output Format

The output contains the answer followed by a newline character.

Sample Input

```
2
2
```

Sample Output

```
10
```

Explanation

Suppose that we label the chocolate bar chunks as follows:

```
|A|B|
|C|D|
```

In this bar, chunk A identified as (1,1), chunk B is identified as (1,2), chunk C is identified as (2,1), and chunk D is identified as (2,2).

There are ten ways this bar could be eaten:

1) A (with B, C, and D at the same time)

2) B (with D), and then A (with C)

3) B (with D), then C, then A

4) C (with D), then A (with B)

5) C (with D), then B, then A

6) D, then A (with B and C)

7) D, then C, and then A (with B)

8) D, then B, and then A (with C)

9) D, then C, then B, then A

10) D, then B, then C, then A

Note: Two other test cases are available if you click on the "Run Code" button.

#19

SAD: Long Way from Home

Make the SADs happy. Show them the way home

SAD: Long Way from Home

Problem Statement

Subterranean Antisocial Demons (SADs) live and roam in a network of connected caves. They move from cave to cave following very strict rules:

- Only one SAD may move at a time.
- Every time a SAD moves, it must move from one cave to an adjacent cave.
- No two SADs can occupy the same cave at the same time. Thus, if a cave is already occupied by a SAD, another SAD cannot move into it.
- A SAD may move out of its home cave to allow another SAD to pass through.
- Every SAD must be in its home cave at the end of the day.

The SADs have been busy roaming all day. Your task is to make the SADs happy. Help them find the fastest way for each one to return to its home.

Input Format

The first line of input will contain an integer t, $1 <= t <= 10$ that indicates how many test cases are present.

Next follow t test cases with the following format:

The first line of the test case is an integer n, $1 <= n <= 15$.

Next come n lines that give the current position of each SAD in the form:

$D_i\ C_j$

where D_i and C_j are integers between 1 and 16 inclusive. This indicates that the SAD with ID D_i is currently in cave with ID C_j. Note that the home cave for a SAD is the cave with an ID equal to the SAD's ID. Since a SAD can only move into an empty cave, there will always be more caves than SADs, and therefore the number of cave ID's will always be larger than the number of SAD ID's.

On the next line is an integer l, $n <= l <= 120$.

Next come l lines that give connection between caves in the form:

$C_i\ C_j$

where both of these IDs are integers between 1 and 20, inclusive. This line indicates that it is possible to get from the cave with ID C_i to the cave with ID C_j, and from the cave with ID C_j to the cave with ID C_i.

Output Format

For each test case, you should output, on a line by itself, a single integer equal to the **minimum** number of moves needed to get **every SAD** from its current location to its home cave.

Note: it will **always be possible for every SAD to reach its home**, and the minimum number of moves needed is never greater than 30.

Sample Input

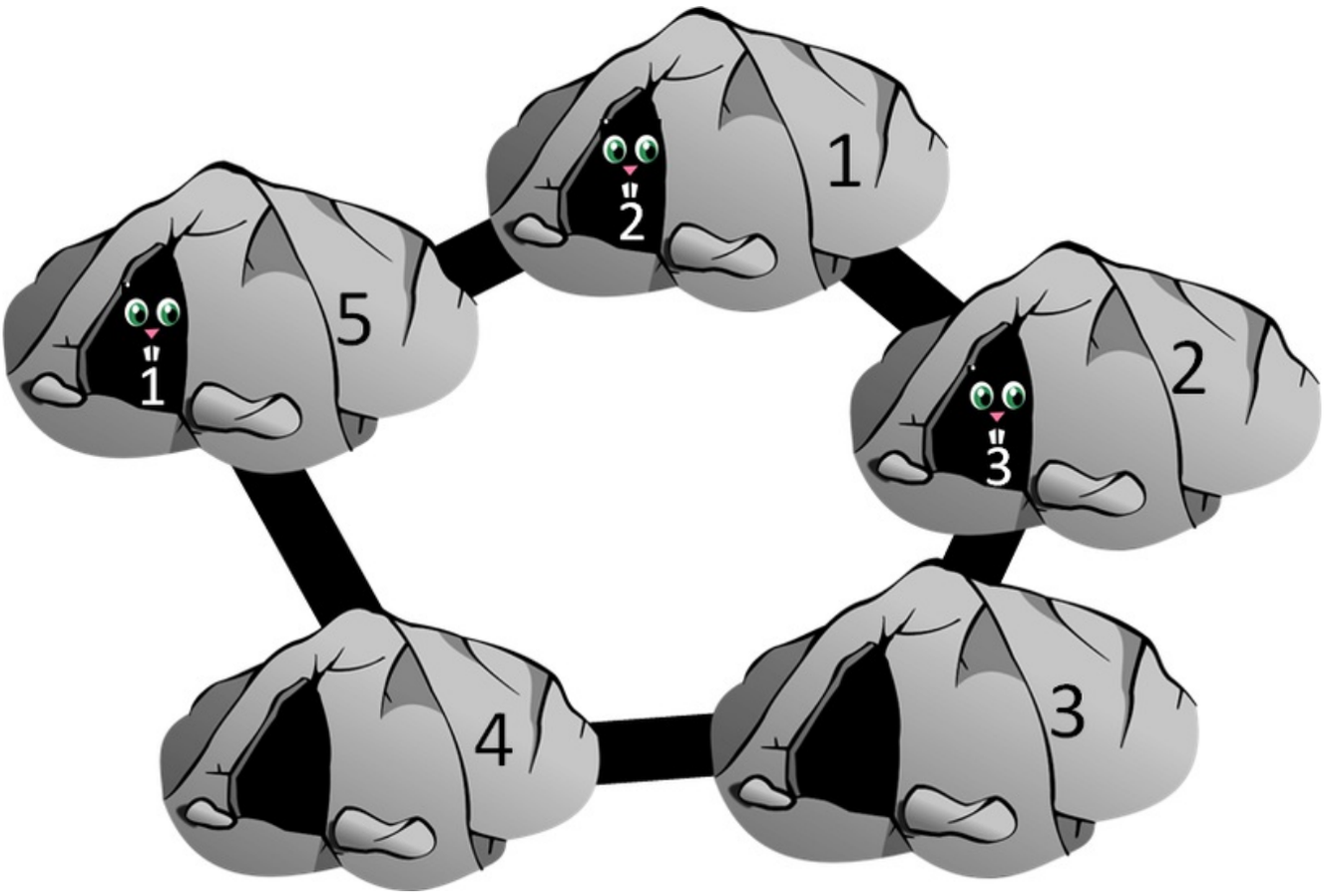
```
1
3
1 5
2 1
3 2
5
1 2
2 3
3 4
4 5
5 1
```

Sample Output

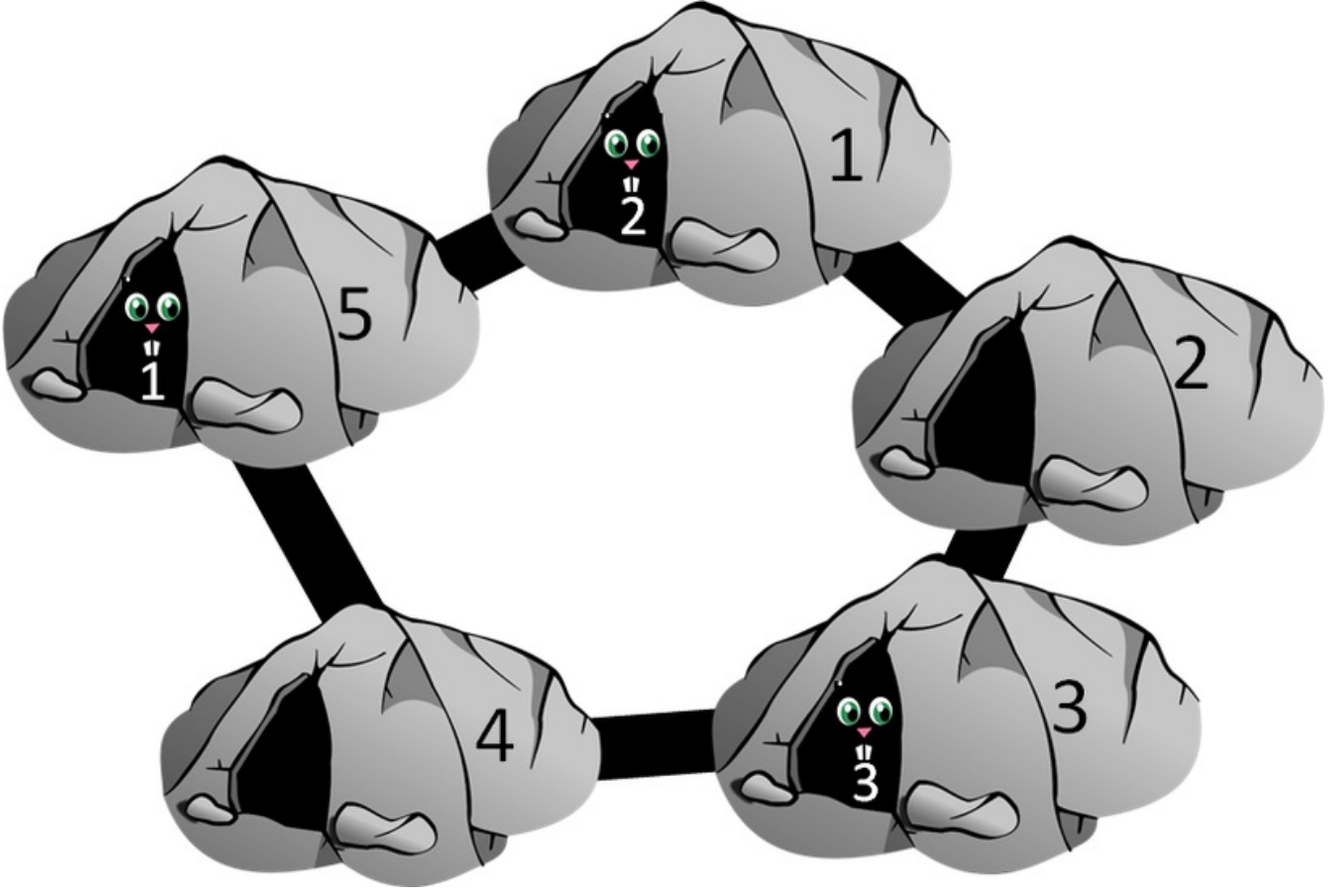
```
3
```

Explanation

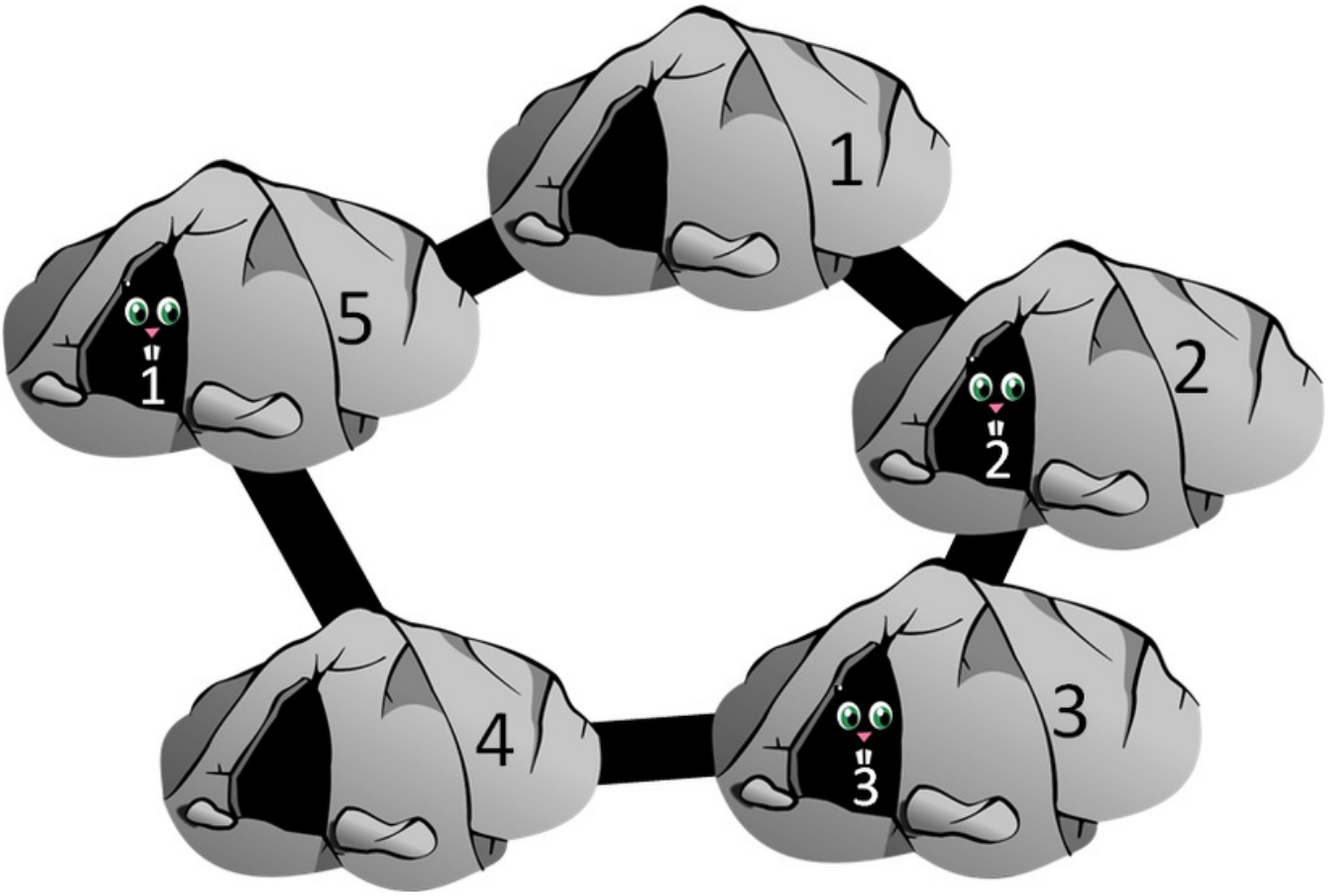
As shown in the next figure, at the beginning of the first test case, SAD 1 is in cave 5, SAD 2 is in cave 1, and SAD 3 is in cave 2. There are two legal moves in this scenario: SAD 1 could move to cave 4 or SAD 3 could move to cave 3.



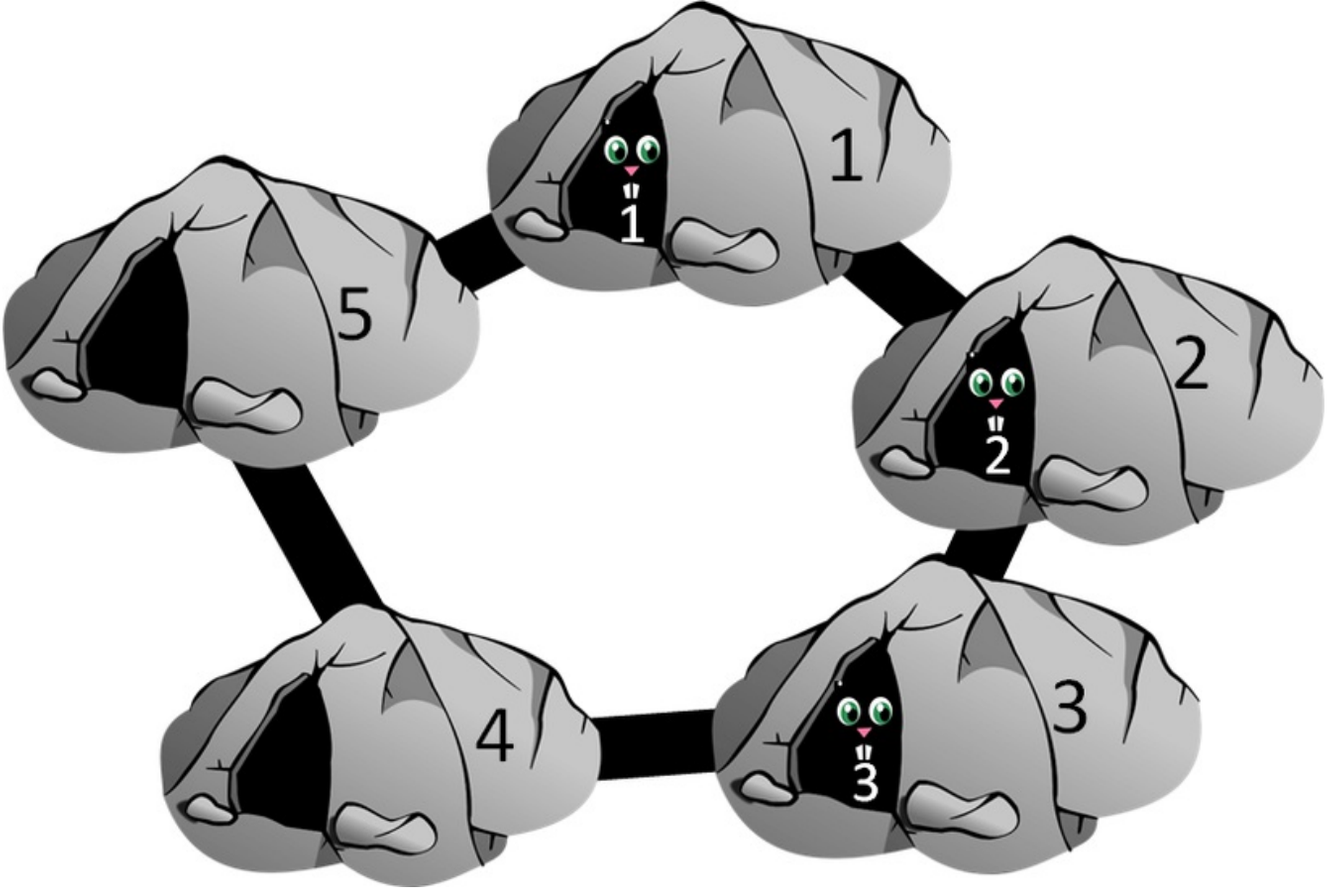
In the first move, then, SAD 3 moves to cave 3. As shown in the next figure, it is home. Now, for the second move, there are four legal moves: SAD 1 could move to cave 4, SAD 2 could move to cave 2, SAD 3 could move to cave 2, or SAD 3 could move to cave 4.



In the second move, then, SAD 2 moves to cave 2. As shown in the next figure, it is also home. For the third move, there are four legal moves: SAD 1 could move to cave 4, SAD 1 could move to cave 1, SAD 2 could move to cave 1, or SAD 3 could move to cave 4.



In the third and final move, SAD 1 moves to cave 1. As shown below, now all of the SADs are home, and it only took three moves!



Images generated from clipart from www.clker.com.

#20

https://www.ieeextremeisbestcontest....

https://Xtrm.org

Shortening in the Real World

Oh, honey, I shrunk the URLs!

Shortening in the Real World

Problem Statement

You are part of a large gaming firm which is looking forward to offer online gaming competitions through live streaming. This means a lot of users will be looking to share their channels links through multiple social media networks (e.g: twitter, facebook, etc.)

Your company wants to implement URL redirection, where you will provide a URL on your website that when requested will redirect browsers to the links provided by users. Due to the burst of popularity of e-sport sites, your company did not plan for the large demand for URLs. The encoded URLs given by the system are very long and difficult to handle by regular users, reducing the usability of your system and service.

Your boss comes to you in a panic, hoping that you will be capable of solving this simple but critical task and help the company help users to handle URLs in an easy and comfortable manner. As a good software engineer with some basic understanding of Computer Science, you devise a way to encode the original encoded URL into a shorter form through a hash function. As you know, there is always the risk that your hash function will have a collision, but it is a good start.

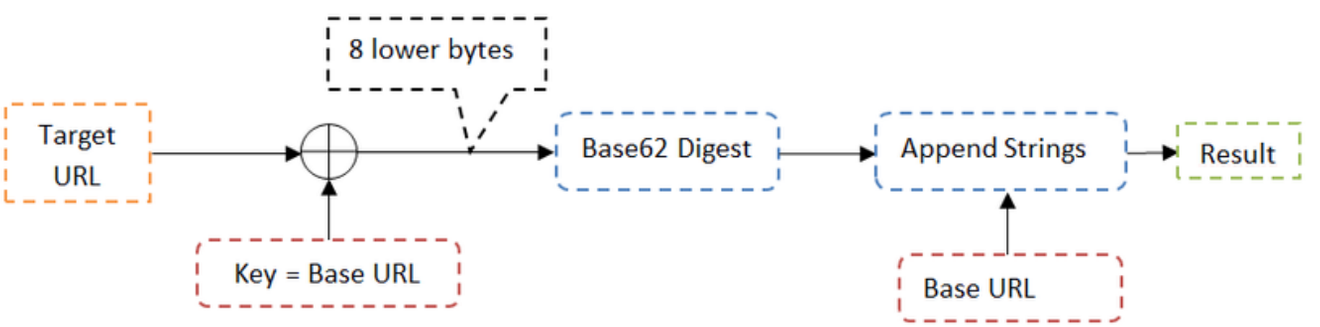
The hash function takes as input the base URL of your company and the target URL you wish to encode. (For an example of this process, see the *Explanation* portion of the sample input, below). As shown in the figure below, the algorithm proceeds in the following steps.

1) An xor cipher is applied to the target URL, using the base URL of your company as a repeating key. Here we perform a bitwise exclusive-or between each byte of the target URL and the base URL. If the target URL is shorter than the base URL of your company, you would truncate the base URL so that the lengths are equal. If the base URL of your company is shorter than the target URL, you would repeat the base URL as many times as needed to make the lengths equal.

2) Take the last 8 bytes of the output from step 1, and convert this to the corresponding unsigned integer. (See the example below for more details.)

3) Encode this unsigned integer using Base62 encoding. In this encoding, you convert the integer to a base 62 number, where the digits 0-9 represent values 0-9, lowercase letters a-z represent values 10-35, and capital letters A-Z represent values 36-61.

4) The encoded url consists of the base URL, a backslash, and the base 62 encoded value produced in the previous step.



Notes:

- You should process the URLs using UTF-8 encoding.

- The URL to be encoded will always be at least 8 characters long.
- The base 62 number should not be padded with extra zeros. For example, the number "62" should be represented as "10", not as "010" or "0010" or "00010". The number "0" would be represented as "0".
- The base URL will never end with a backslash.

Input Format

The first line of input will contain the base URL of your company.

The second line will contain a number *n* (1 <= *n* <= 1000) which will indicate the number of URLs you will need to encode.

The following *n* lines will contain target URLs to encode.

Output Format

The output should be *n* lines, where each line will correspond to an encoded URL.

Sample Input

```
http://www.ieee.com
2
http://www.ieee.org/xtreme
http://www.ieee.org/membership_services/membership/young_professionals/index.html
```

Sample Output

```
http://www.ieee.com/SHPQ4gzW1Y
http://www.ieee.com/Btazwa9mke
```

Explanation

Consider the first target URL to be encoded: `http://www.ieee.org/xtreme`.

We start by finding the UTF-8 encoding of the base URL `http://www.ieee.com` and the target URL.

```
target URL = [0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77, 0x77, 0x77, 0x2e, 0x69, 0x65, 0x65, 0x65, 0x2e, 0x6f, 0x72, 0x67, 0x2f,
0x78, 0x74, 0x72, 0x65, 0x6d, 0x65]
base URL   = [0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77, 0x77, 0x77, 0x2e, 0x69, 0x65, 0x65, 0x65, 0x2e, 0x63, 0x6f, 0x6d]
```

Now we apply the exclusive-or cipher, repeating the bytes in the base URL so that the two strings are the same length:

```
   [0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77, 0x77, 0x77, 0x2e, 0x69, 0x65, 0x65, 0x65, 0x2e, 0x6f, 0x72, 0x67, 0x2f, 0x78, 0x74,
0x72, 0x65, 0x6d, 0x65]
xor
   [0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77, 0x77, 0x77, 0x2e, 0x69, 0x65, 0x65, 0x65, 0x2e, 0x63, 0x6f, 0x6d, 0x68, 0x74,
0x74, 0x70, 0x3a, 0x2f, 0x2f]
=
   [0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc, 0x1d, 0x0a, 0x47, 0x0c,
0x00, 0x02, 0x5f, 0x42, 0x4a]
```

Next we take the last 8 bytes of this number and convert it to an unsigned integer:

0x0a470c00025f424a = 740,573,857,905,066,570

Next we convert this number from base 10 to base 62 to get: `SHPQ4gzW1Y`

We then append this number to the base URL to produce the output: `http://www.ieee.com/SHPQ4gzW1Y`

#21

Communities

If it's not online it didn't happen. Show us your photos :)

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Communities

Problem Statement

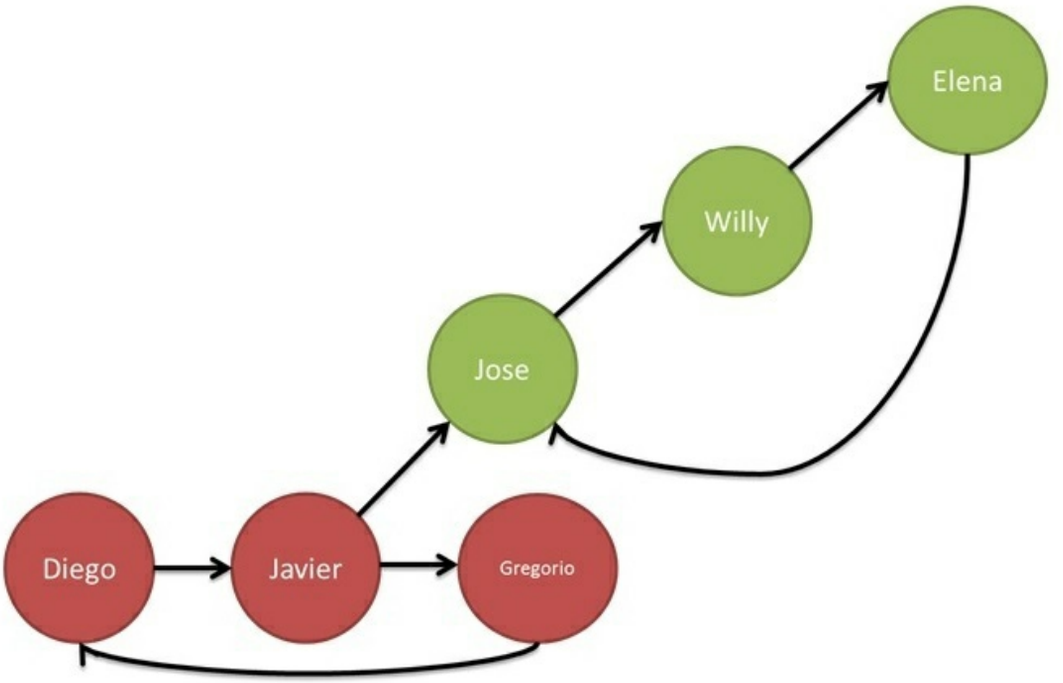
Social media networks and the amount of information they aggregate is astonishing. With so much information, new patterns and interactions of human society can be identified.

In our social network, the relationships model the flow of information, and they are directed. Alice may subscribe to Bob's newsfeed, while Bob does not subscribe to Alice's. Moreover, the flow of information in our network is such that a person can see the newsfeeds of all people who could reach the person following along a path in the network. Suppose, then, that Alice subscribes to Bob's newsfeed, Bob subscribes to Chuck's newsfeed, and Chuck subscribes to Dave's newsfeed. This would correspond to a simple linear graph:

Alice <- Bob <- Chuck <- Dave

Then Dave would be able to read his own news items only; Chuck would be able to read news items posted by either Dave or himself; Bob would be able to read news items posted by either Chuck, Dave or himself; and Alice would be able to read everyone's news items. Note that everyone can read their own newsfeed.

We are interested in the defining a community metric for our social network. We define a community as a group of people who are able to see all news items posted by any member of the group. As an example, in the figure below, there are two communities, each shown in a different color.



Note that in the community shown in green above, Jose, Willy, and Elena can all read each other's posts. While Jose, Willy, and Elena can also read Javier's news items. However, Javier cannot read news items from Jose, Willy, or Elena, and is therefore not included in their community.

Your task is to identify the sizes of these communities from biggest to smallest.

Input Format

The first line of input will contain two space separated integers: the total number of people that devise the social network, *n* (1 <= *n* <= 10000) and *m*, the number of communities for which you should print the size. The following lines will contain a directed relationship between 2 people. If the line reads "Jon Peter",

then Peter subscribes to Jon's news feed, and the relation is Jon -> Peter.

The word "END" will appear on a line by itself after the list of relationships.

All of the names are strings containing fewer than 50 characters.

Output Format

The output consists of *m* lines, where each line will correspond to the size of a community from biggest to smallest. If there are fewer than *m* communities, after outputting the size of all existing communities, output lines containing “Does not apply!” for the missing values.

Sample Input

```
6 2
Jose Willy
Willy Elena
Elena Jose
Diego Javier
Javier Gregorio
Gregorio Diego
Javier Jose
END
```

Sample Output

```
3
3
```

Explanation

This input corresponds to the graph described in the problem statement above.

Note that two additional sample inputs are available if you click on the "Run Code" button.

#22

Pattern

When you have a signal, everything looks like a pattern!

Pattern 3

Problem Statement

Vangelis the bear received a digital signal pattern generator that his brother Mitsos built. The generator produces a signal that is encoded using the Latin alphabet. Vangelis starts the generator for some time and records the signal generated. He wants to study the sample he received and try to identify the smallest pattern that the generator could be using to generate the sample.

Your task is to help Vangelis by writing a program that will calculate the length of the smallest valid pattern.

Input Format

The input is made up of multiple test cases.

The first line contains an integer T ($1 <= T <= 10$), the number of test cases in this file.

Each line contains an encoded signal. The signal is encoded using the small letters of the Latin alphabet. The length of a signal is between 1 and 10^6 characters, inclusive.

Vangelis has started the recording at the beginning of a pattern, so each line begins with the first character in the pattern. His recording lasts for at least one pattern length, but the length of the recording **may not be an exact multiple of the pattern length**.

Output Format

There must be T lines of output and each line will contain a single non-negative integer number, the length of the minimum valid pattern.

Sample Input

```
6
abab
abababababababababab
abababababab
abc
aaaaaa
aabaabbaabaabbaabaabbaabaab
```

Sample Output

```
2
2
2
3
1
7
```

Explanation

The repeating patterns in each of the test cases are:

```
ab
ab
ab
abc
a
aabaabb
```

#23

Land

Help us design our XtremeLand, the land were xtreme things happen!

Land

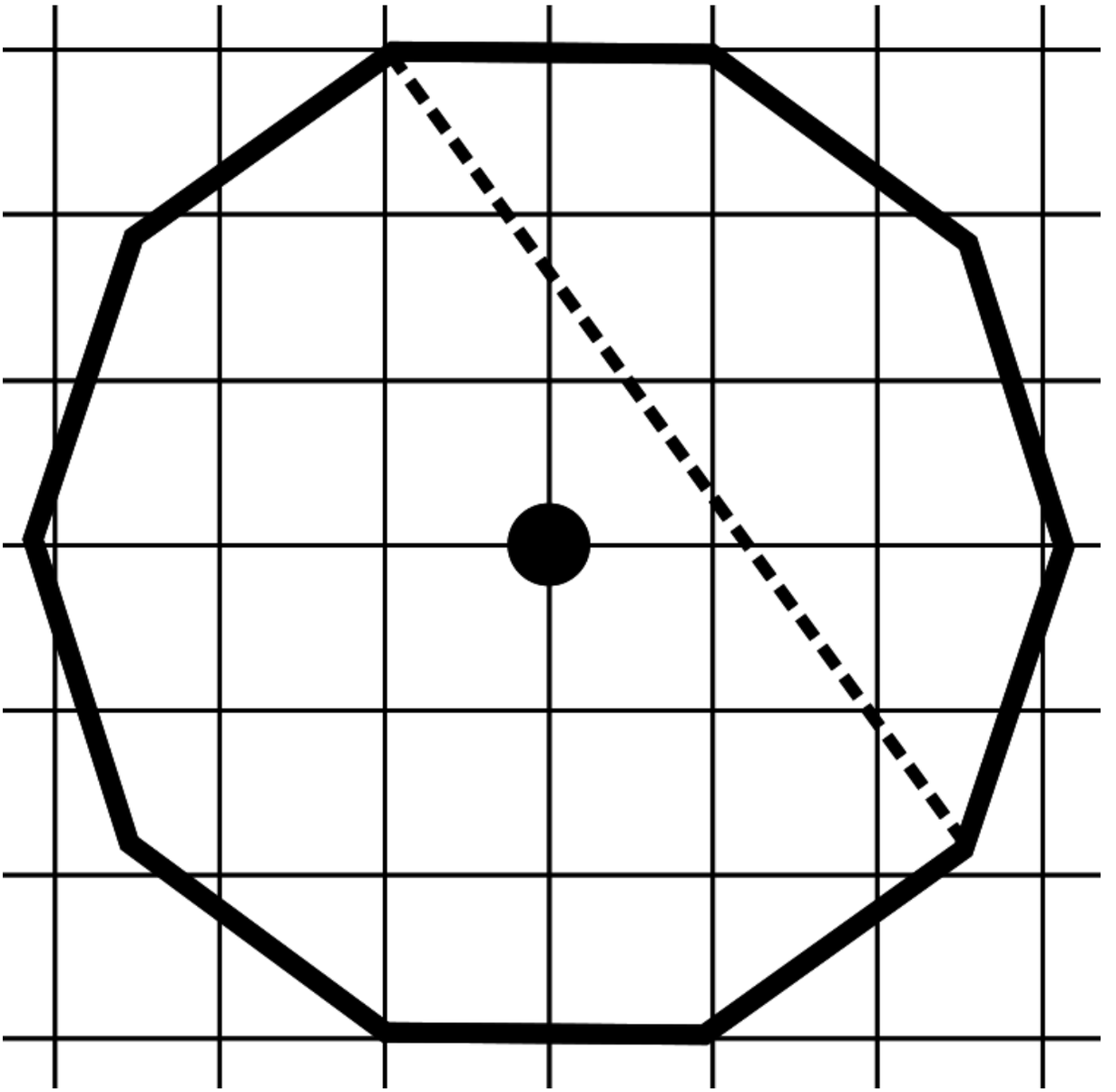
Problem Statement

Vangelis the bear bought a real property inside a forest where he wants to build his new corporate headquarters. The shape of the land is a strictly convex polygon. This means that the shape is a polygon with all its interior angles less than 180°. All vertices of the shape are on a plane, whose locations are given in a Cartesian coordinate system.

In the plot, there are some really old trees that Vangelis wants to protect. To do so, Vangelis decided that he will separate his land into two parts. The first part, where there must not be any trees, will be used for the headquarters building. The second part, where the ancient trees reside, will be preserved as a park.

Being a scientist, he decided to make the separation using a diagonal of the polygon. Your task is to help Vangelis compute the maximum possible land that he can use to build his headquarters.

Note: A tree cannot appear on the border between the regions. Thus, in the image below, the maximum area for the headquarters *is not equal* to one half the area of the land.



Input Format

The input is made up of multiple test cases.

The first line contains an integer T ($1 <= T <= 10$), the number of test cases in this file.

The following repeats T times:

- The first line in the test case contains an integer N ($4 <= N <= 300000$), the number of vertices that compose the land shape.
- Each of the following N lines contain two single-space-separated integers X and Y ($-10^9 <= X, Y <= 10^9$), the coordinates of a vertex. The vertices are given in a counter-clockwise order going around the boundary of the land.
- The line after contains an integer M ($0 <= M <= 300000$), the number of trees inside the land.
- Each of the following M lines contain two single-space-separated integers I and J, the coordinates of a tree inside the land. None of the trees will be placed on an edge of the land.

Output Format

There must be T lines of output and each will contain a single non-negative real number, the maximum possible area that can be used for the headquarters in the specific test case. The numbers should always be rounded to two decimal places, and they should always display these two decimal places even if the final value is an integer. In the cases that there is no available land to be used, your program should write 0.00.

Sample Input

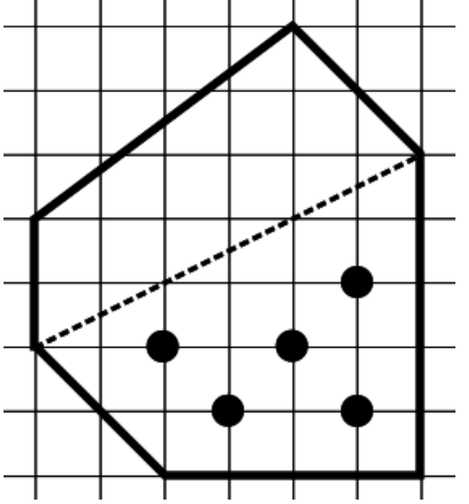
```
1
6
2 0
6 0
6 5
4 7
0 4
0 2
5
2 2
3 1
4 2
5 1
5 3
```

Sample Output

```
13.00
```

Explanation

The image below represents the optimal division of the land resulting in clear space for headquarters of 13 m^2.



Note: An additional test case is available if you click on the "Run Code" button.

#24

e = 2.718281828459045070 51

That's So Characteristically Euler!

Euler, we've got your number!

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE computer society

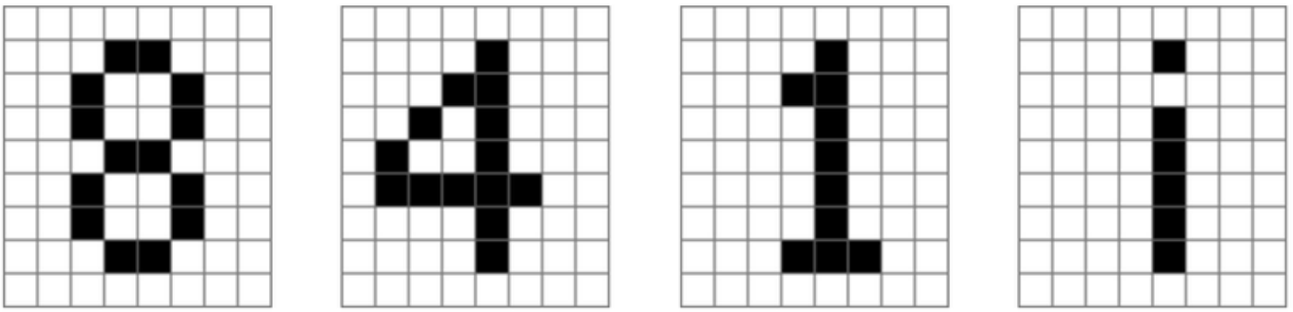
IEEE

That's So Characteristically Euler!

Problem Statement

Topology is a branch of mathematics that examines the shapes and structures of objects. The *Euler characteristic* is a topological label that we can determine for an item or collection of items. For a group of two dimensional objects, one way to define the Euler characteristic is the number of connected objects minus the number of holes in the objects.

The Euler characteristic has useful applications in 2D image recognition and classification. Consider the following black and white representations of three digits and a letter and their corresponding Euler characteristics.



As you can see, the Euler characteristic can give a lot of information about what a given image might or might not represent. Note that, in this problem, we consider filled-in regions that touch diagonally to be part of the same connected region; however, holes that only touch diagonally are considered to be separate. Your task is to compute the Euler characteristic (connected regions minus holes) for a black and white image similar to those above.

Input Format

Each input contains multiple test cases. The first line of input contains an integer t, $1 <= t <= 10$.

Following this line are t test cases, each with the following format:

- The first line of the test case contains an integer n, giving the height of the image.
- The second line of the test case contains an integer m, giving the width of the image.
- The rest of the test case contains n lines of text. These n lines of text contain m characters, each of which is either the letter 'X' or the letter 'O'. An 'X' represents a black pixel, and an 'O' represents a white pixel.
- The possible ranges for n and m are $1 <= n <= 1{,}000$ and $1 <= m <= 1{,}000$. The input will always be correctly formatted.

Output Format

The correct output for each input is a single integer per test case (each integer on its own line) which gives

the Euler characteristic for each input image.

Sample Input

```
1
4
4
OOOO
OXXO
OXXO
OOOO
```

Sample Output

```
1
```

Explanation

The sample input contains a single image. The image contains a single connected square region surrounded by empty space.

Thus, 1 region - 0 holes = 1.

#25

Finite Domain Constraints

Back 2 school: Let's do some basic math.... :)

Finite Domain Constraints

Problem Statement

A finite domain constraint is a linear equality or inequality over integer variables. Each variable has a known and finite domain (set of possible values). A finite domain constraint is satisfiable if each variable can be assigned a value from its domain in a way that makes the equality or inequality true. For example, given the finite domain constraint:

$(X + Y) = 5$

with the domain of X as {1, 2, 3, 4} and the domain of Y also as {1, 2, 3, 4}, the constraint is satisfied by assigning 2 to X and 3 to Y. Note that the constraint can also be satisfied by assigning 3 to X and 2 to Y, 1 to X and 4 to Y, and 4 to X and 1 to Y. Hence, there are 4 distinct variable assignments that satisfy the constraint. If we change the constraint to the inequality:

$(X + Y) < 5$

with the same domains, there are now 6 distinct variable assignments that satisfy the constraint: ($X = 1$, $Y = 1$), ($X = 1$, $Y = 2$), ($X = 1$, $Y = 3$), ($X = 2$, $Y = 1$), ($X = 2$, $Y = 2$), and ($X = 3$, $Y = 1$).

Given a finite domain constraint and domains for each variable, your task is to determine how many distinct variable assignments satisfy the constraint.

Input Format

The input is made up of multiple test cases. Each test case ends with a 0 on a line by itself.

The first line in each test case is an integer n, $1 <= n <= 10$. n is the number of variables in the finite domain constraint.

The next n lines are each of the form:

[Variable_Name] [Low] [High]

where:

- [Variable_Name] is the variable name
- [Low] is the lower bound of the variable's domain (inclusive)
- [High] is the upper bound of the variable's domain (inclusive)

For example:

```
X 1 4
```

means that the domain of X is {1, 2, 3, 4}. Variable names are not repeated within the same test case.

The next line of each test case is a finite domain constraint over those variable names.

Notes:

- The lower and upper bounds are integers from 1 to 10 (inclusive).

- The constraint will contain exactly one occurrence of either = or <. Note that the operator <= will never appear in a constraint.
- The constraint can additionally contain parentheses, +, -, *, variable names and integer constants.
- The constraint will be fully parenthesized so that precedence (order of operations) is not needed.
- Each distinct symbol in the constraint will be separated from other symbols by exactly one space.
- Each variable occurs exactly once in the constraint.
- The number of integer constants is less than or equal to the number of variables plus one, and each constant is between -100 and 100 (inclusive).
- When the * operator is used in a constraint, one operand will be a constant and the other will be a variable.
- You can assume that the constraint is syntactically correct.

Output Format

For each test case, your program should output, on a line by itself, the number of distinct variable assignments that satisfy the finite domain constraint with the given variable domains.

Sample Input

```
2
X 1 4
Y 1 4
( X + Y ) = 5
0
3
X 1 3
Y 2 4
Z 1 4
( ( X + Y ) - ( 2 * Z ) ) < 5
0
4
W 2 5
X 1 3
Y 2 9
Z 1 4
( ( X + ( Y - 3 ) ) + ( -2 * Z ) ) = ( ( 5 * W ) - 10 )
0
```

Sample Output

```
4
35
15
```

Explanation

The first test case in the sample input corresponds to the first example given in the problem statement.

#26

Blocks Game

Blocks blocking blocks

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Blocks Game

Problem Statement

In this problem, you will solve a puzzle from the Secret Society Hidden Mystery Game by G5 Games. A typical board for this game is shown here:

Each block can only move in one dimension: vertical blocks can only move vertically and horizontal blocks can only move horizontally. The goal of the game is to slide the yellow block out of the frame through the hole in the frame on the right side. To do that, you have to slide the rest of the blocks out of the way. In addition, for this problem, you must do it in the smallest number of moves possible.

Here are some details to constrain our game:

- Moving a block through two spaces counts as two moves.
- The task is just to find the minimum number of moves required to win the game – the sequence of specific moves is not important.
- The board is in form of a square.
- Once the yellow block is located right next to the exit, you have won the game. You do not need to move the yellow block off the board.
- The blocks can only move inside the board, and movement of any block outside the board is not permitted.
- The game is always winnable, in a finite number of moves.

Input Format

Each run of the program will process a sequence of test cases each containing an instance of the game.

The first line of input contains a number T. T specifies the number of test cases in the input.

Each test case is specified as follows:

On the first two lines of the test instance is an integer Y, representing the column with the leftmost side of the yellow block and N, giving the dimensions of the board. Since the board is always square, N is the number of rows and the number of columns of the board. Hence, the board coordinates can be considered as an N x N matrix (i.e. the top left corner of the board is row 0 and column 0). The yellow block is always located in the third row (i.e. it is in the row labelled 2).

The third line of the test case contains an integer *B* representing the number of brown blocks in the problem.

The next *B* lines give the location of each of the brown blocks in the following format:

S D R C

Where

- *S* is an integer give the number of spaces the block covers. A block will always fit in the board.
- *D* is either 'V' or 'H' designating whether the block is horizontal or vertical
- *R* and *C* are the row and column, respectively, of the top left square of the block

Constraints:

- 1 <= *T* <= 5
- 0 <= *Y* < *N* - 2
- 4 <= *N* <= 6
- 1 <= *B* <= 12
- The maximum number of unique board configurations that can be reached in any test instance is 250,000.

Output Format

The output must be one integer for each instance of the problem on a separate line giving the minimum number of moves required to win the game.

Sample Input

```
1
0
6
3
3 V 0 2
3 V 2 5
2 H 5 2
```

Sample Output

```
9
```

Explanation

The input specification corresponds to the image below.

This game is winnable in 9 moves. For example, one possible sequence of moves that wins in the shortest number of moves is:

1) Move the horizontal brown block one square to the right.

2-4) Move the left vertical block down three squares. Note that this counts as three moves.

5-7) Move the yellow block to the right three squares.

8) Move the right vertical block down one square.

9) Move the yellow block right one square so that it is next to the exit.

#27

Would Be... Could Be... BST!

In a world where trees grow down, only some can claim the title of BST.

Would Be... Could Be... BST!

Problem Statement

You are given an ambiguous description of one or more binary trees. Each key value is a non-empty string of lowercase letters, and each key occurs in only one node in the collection. You must decide if the description that is given must have been derived from a single binary search tree, could have been derived from a single binary search tree, or could not have been derived from a binary search tree. Please refer to http://en.wikipedia.org/wiki/Binary_search_tree for background information on binary search trees.

Note: The default time limits for all languages have been doubled for this program.

Input Format

The input is made up of multiple test cases. Each test case begins with an integer N, with 1 <= N <= 50,000. Next come N lines each describing an edge in the collection of binary trees:

[key1] [key2]

where [key1] and [key2] are strings corresponding the key values of two nodes, *node1* and *node2*, respectively, where *node1* is either the left child of *node2* OR *node2* is the right child of *node1*.

The last line of input following all of the test cases consists of the number 0 on a line by itself. The total number of edges in all test cases in a single input file will be less than or equal to 500,000.

Output Format

For each test case, you will output a single string followed by the newline character:

- If the description given in the test case must correspond to a binary search tree, you should output “BST!”.
- If the description given in the test case corresponds to a binary search tree in some instantiations but not in others, you should output “BST?”.
- If the description given in the test case cannot describe a binary search tree, you should output “!BST”.

Sample Input

```
2
alpha beta
beta gamma
2
alpha delta
alpha gamma
2
alpha beta
delta gamma
1
beta alpha
0
```

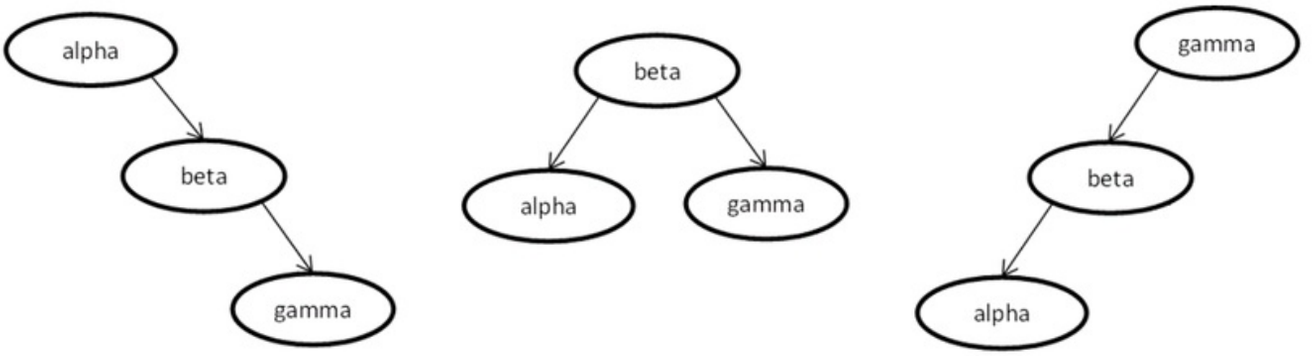
Sample Output

```
BST!
```

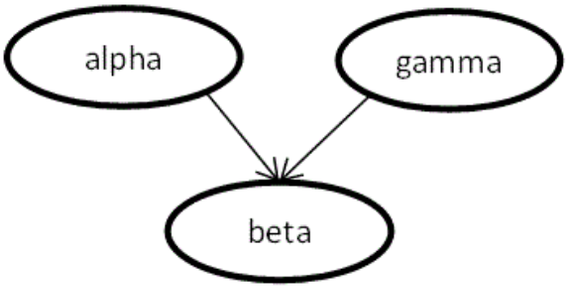
```
BST?
!BST
!BST
```

Explanation

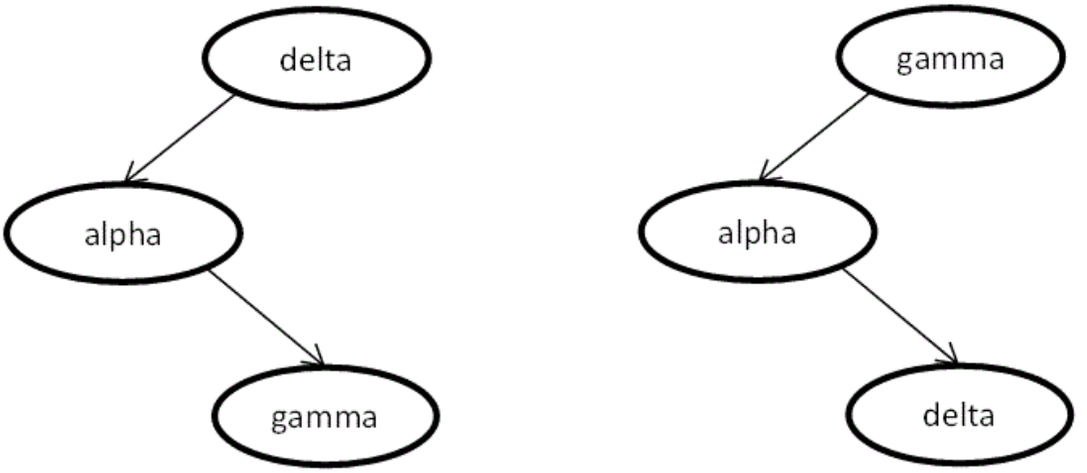
The first test case could have been derived from three different configurations, which are shown below. Note that all of these configurations are binary search trees.



NOTE: Since each test case is a description of one or more binary trees, the description for the first test case could not have come from a configuration like the one below which is not a binary tree:



The second test case describes one of the configurations below. The tree on the right is a binary search tree because it fulfills the binary search tree property, (i.e. the key in each node of the tree is greater than all the keys stored in the left sub-tree, and smaller than all the keys in the right sub-tree). However, the configuration on the left is not a binary search tree because the key value stored at the root of the tree is not smaller than all the values stored in the left sub-tree (i.e. value "gamma" of the left subtree is greater than "delta").



The third test case describes a forest of two trees, which cannot be a BST.

The last test case cannot be a BST.

#28

Alice and Bob play Sheldon's Favorite Game

Sometimes "Rock, Paper, Scissors" just will not cut it.

Alice and Bob Play Sheldon's Favorite Game

Problem Statement

Rock, Paper, Scissors, Lizard, Spock is a game, invented by Sam Kass and Karen Bryla, that extends the Rock, Paper, Scissors game. It is a favorite of the character Sheldon from the TV Show, *The Big Bang Theory*.

The rules of the game are as follows. Each player chooses a shape from the set {*Rock*, *Paper*, *Scissors*, *Lizard*, *Spock*}. If the two players choose the same shape, they tie. Otherwise, the winner is the one with the shape listed first in the following set of rules:

```
Scissors cuts Paper
Paper covers Rock
Rock crushes Lizard
Lizard poisons Spock
Spock smashes Scissors
Scissors decapitates Lizard
Lizard eats Paper
Paper disproves Spock
Spock vaporizes Rock
(and as it always has) Rock crushes scissors
```

Below is a graphical representation from *The Big Bang Theory's* wiki:

Alice and Bob also enjoy this game, and they have decided to play a series of them. Each has a very specific strategy that they follow.

Alice's strategy is as follows:

1) If she wins a game, she keeps the same shape.

2) If she ties, she chooses a shape from one of the two that would beat her current shape. Of these two, she chooses the one that beats the other. For example, if she has tied when choosing *Rock*, her options are *Paper* and *Spock*. Since *Paper* beats *Spock*, she chooses *Paper*.

3) If she loses, she chooses a shape from one of the two that would beat her opponent's current shape. Of these two, she chooses the one that beats the other. For example, let's say she has lost by choosing *Rock*, when her opponent chose *Paper*. She will then choose from *Scissors* or *Lizard*. Since *Scissors* beats *Lizard*, she chooses *Scissors*.

Bob's strategy is as follows:

1) Every other turn, he chooses *Spock*.

2) If he won the previous turn when playing *Spock*, he chooses *Rock*.

3) If he tied the previous turn when playing *Spock*, he chooses *Lizard*.

4) If he lost the previous turn when playing *Spock*, he chooses *Paper*.

Your task is to write a program that evaluates a series of games between Alice and Bob.

Input Format

The first line of input contains a single integer *t*, 1 <= *t* <= 50, containing the number of test cases in the input.

Then come *t* lines each describing a test case, made up of a series of games for you to evaluate. These lines have the following format:

```
[AliceShape] [BobShape] [n]
```

Where

- [AliceShape] is the shape that Alice will choose in the first game of the series.
- [BobShape] is the shape that Bob will choose in the first game of the series.
- [n] is an integer, 1 <= n <= 10^{18}, indicating how many games Alice and Bob will play in the series.

Output Format

Output will consist of a single line in the appropriate one of the following forms:

```
[Player] wins, by winning [WinGames] game(s) and tying [TieGames] game(s)

Alice and Bob tie, each winning [WinGames] game(s) and tying [TieGames] game(s)
```

where

- [Player] is the name of the player with more wins (either "Alice" or "Bob")
- [WinGames] is the number of games won either by the winner or, in the case of a tie, by each player
- [TieGames] is the number of games in which the players tied

Notes:

- The output is case sensitive. The player names, for example, must be either "Alice" or "Bob". Neither "alice" nor "BOB" will be acceptable.
- The words are separated by a single space, and there are no spaces before the first word in the line, nor after the last word in the line.

Sample Input

```
2
Rock Spock 4
Paper Paper 1
```

Sample Output

```
Bob wins, by winning 2 game(s) and tying 1 game(s)
Alice and Bob tie, each winning 0 game(s) and tying 1 game(s)
```

Explanation

There are two test cases in this input:

Test Case 1

In the first game, Bob wins, since *Spock* vaporizes *Rock*.

Bob won when choosing *Spock* so he chooses *Rock*. Alice lost, so she chooses from *Paper* and *Lizard*, both of which beat Bob's last choice of *Spock*. Since *Lizard* beats *Paper*, she chooses *Lizard*.

In the second game, then, Bob wins again, because *Rock* crushes *Lizard*.

Bob did not play *Spock* last turn, so he chooses *Spock* next. Alice lost, so she chooses from *Paper* and *Spock*, both of which beat Bob's last choice of *Rock*. Since *Paper* beats *Spock*, she chooses *Paper*.

In the third game, Alice wins, since *Paper* disproves *Spock*.

Bob lost when choosing *Spock* so he chooses *Paper*. Alice won, so she continues playing *Paper*.

In the fourth game, they tie by both choosing *Paper*.

Test Case 2

This test case consists of a single game in which both players play *Paper*.

#29

Sleep, Eat, Drink, Code

What else is there?

Eat, Sleep, Drink, Code

HackerRank

Problem Statement

Alice has recognized that strategy is key for success in IEEEXtreme. Your task is to help her by writing a program that will calculate the maximum score that she can earn in marathon competitions like Xtreme.

Alice starts the competition with a certain energy level. She can only attempt a problem if her energy level prior to starting the problem is greater than or equal to the energy required by the problem. Furthermore, if she attempts to solve the problem, her energy level after the hour is reduced by the energy required by the problem.

Alice wants you to assume that every hour a problem is released, and she can make the following decisions:

- Attempt to solve the problem. She is able to accurately predict how many points she will earn by attempting the problem.
- Skip the problem and sleep. Note that she will not come back to this problem later. She will gain a fixed amount of energy by doing so.
- Drink a caffeinated cola and attempt the problem, if she has drinks remaining. She will gain a fixed amount of energy *immediately*. As long as her resulting current energy level is greater than or equal to the energy level required by the problem, she can attempt to solve it. As usual she will expend the energy required to solve the problem. In addition, as the caffeine wears off, she will lose a certain amount of energy units exactly two hours later.

Notes:

- It is ok for her energy level to become negative after losing the points due to the cola consumption. However, she will need to boost her energy by sleeping or drinking additional cola before she will be able to solve a problem.
- She can only drink one cola per hour.
- If she drinks a cola, she must attempt the problem.
- For each hour that she sleeps and skips a problem, she gains the fixed amount of energy. Thus, if she sleeps for two consecutive hours, she will gain twice as much energy as if she slept for one hour. If she sleeps for three consecutive hours, she will gain three times as much energy, etc.

Input Format

The first line of input contains an integer k, $1 <= k <= 20$, which indicates how many test cases are present.

Each test case then has the following format. The first line of the test case consists of the following:

```
[Hours] [Energy] [Sleep] [DrinkCount] [Drink] [Crash]
```

Where

- [Hours] gives the length of the contest in hours, 1 <= [Hours] <= 168. (Alice envisions the day when Xtreme is a week-long contest!)

- [Energy] is Alice's energy level at the beginning of the contest, 0 <= [Energy] <= 10^7.
- [Sleep] is the amount of energy that Alice gains by skipping a problem and sleeping, 1 <= [Sleep] <= 10^6.
- [DrinkCount] is a count of colas that Alice has at the start of the contest, 0 <= [DrinkCount] <= 24.
- [Drink] is the initial boost that Alice receives from drinking a cola, 1 <= [Drink] <= 10^6.
- [Crash] is the amount of additional energy that Alice loses two hours after drinking a cola, 1 <= [Crash] <= 10^6.

Then there follow [Hours] lines, each describing a problem, and listed in the order in which the problem is released, i.e. the problem on the first line is released at the start of the contest, the second problem is released one hour later, the third problem is released an hour after that, etc. These lines have the following format:

```
[EnergyRequired] [Points]
```

Where

- [EnergyRequired] is an integer equal to the amount of energy that Alice will expend in attempting to solve the problem, 1 <= [EnergyRequired] <= 10^7.
- [Points] is equal to the points that Alice will earn if she attempts the problem. [Points] will be equal to an integer chosen from the following set {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.

Output Format

For each test case, output on a separate line, a single integer equal to maximum amount of points Alice can earn in the contest.

Sample Input

```
1
4 5 16 2 7 8
10 100
6 50
15 20
3 10
```

Sample Output

```
160
```

Explanation

There is only one test case in the sample input. For this test case, the contest lasts 4 hours, Alice's initial energy level is 5, she gains 16 energy units by sleeping, she has 2 colas, she gains a boost of 7 energy units by drinking a cola, but then loses 8 energy units after two hours.

In order to solve the first problem, she drinks a cola in order to boost her energy level to 12. She can then solve the first problem, and earns 100 points, by doing so. The first problem takes 10 energy units to solve. Note that she will be penalized by 8 energy units when the third problem is released as the caffeine wears off.

At the start of the second hour, then, she has 100 points, and an energy level of 2. In order to solve the second problem, she drinks another cola. This raises her energy level to 9. She then solves the problem, earning 50 more points. She expends 6 energy units solving the problem. Note that she will lose an additional 8 energy units as the caffeine wears off prior to the release of the fourth problem.

At the start of the third hour, she has 150 points. Her energy level is depleted both by solving the previous problem and as the caffeine from the first cola wears off. Thus her energy level is -5.

She decides to sleep instead of solving the third problem. Her energy level is boosted to 11 by sleeping but reduced to 3 as the caffeine from the second cola wears off.

She then solves the fourth problem and earns 10 more points.

This sequence of decisions maximizes her point total, so your program should output 160, the total number of points that she earned.

The Results

IEEEXtreme Stats

Teams logged in

2317

Submissions made

48220

Teams with submissions

2028

Compile-tests made

263315.

Leaderboard

Filter by Select filter

Rank	User	Score	Time	Country
1	TeamName	2434.98	20009:17	
2	EPFL1	2434.42	21661:09	
3	Powerhouse	2114.11	19872:29	
4	DELAPAN3gp	2096.33	20533:26	
5	IAmSuaAndorinhaM...	2085.13	19674:37	
6	AingeCP	2078.64	20338:06	
7	JUIITCoders	2008.02	20168:57	
8	Benchwarmers	2004.09	19834:59	
9	HybridCode	1931.30	19052:11	
10	AcarajeComFarofa	1918.24	21223:28	
11	SorryHybridCode	1871.42	20396:01	
12	HackRush	1868.02	20057:17	
13	UPPG	1830.85	16735:06	
14	RJK	1820.42	15669:41	
15	HelloWorld	1780.16	18839:58	
16	CUCPMeowMeow	1764.34	18904:04	
17	EarlGrey	1744.00	14900:24	
18	MonashMS	1728.03	22279:49	
19	GoodGame	1719.14	24610:42	
20	Fate	1713.42	25928:54	
21	Kadkhoda	1711.96	17517:53	
22	GantengGantengKo...	1700.06	18840:56	
23	CoDeFX	1692.57	19098:01	
24	TnTWizard	1681.11	19695:48	
25	PlasticSpirit	1674.17	23099:48	

26	KSUNetSE	1672.22	22857:22	
27	1000KB	1654.52	20575:45	
28	VirtualVagrants	1651.24	20712:15	
29	biubiubiubiubiu	1648.94	27414:19	
30	OmeletteDuFromag...	1639.41	18166:30	
31	BB8	1635.06	14859:26	
32	Noop	1621.79	17171:32	
33	EnBuyukYumruk	1613.82	23586:28	
34	SKT1	1613.57	21995:29	
35	CSharpAugmentedT...	1588.72	22019:59	
36	LocosPorTequila	1574.93	19485:32	
37	ITBHitzzz	1570.75	19194:15	
38	OhMyGod	1557.93	23859:07	
39	LikhaWTarneeb	1552.00	16418:37	
40	zEXYkLAP4OVCI2po...	1541.77	13056:54	
41	AC	1540.71	19898:38	
42	TheRealDeal	1531.82	25213:54	
43	IdeaLight	1508.88	14990:15	
44	CoolCoders	1507.01	29759:24	
45	CacheMoney	1500.76	18284:51	
46	HeapHeapArray	1492.29	27275:13	
47	Atsisveikinimas	1472.45	24944:19	
48	Byte	1463.15	26467:06	
49	Codigu	1463.11	18914:43	
50	NullPointerExcep...	1456.14	28006:33	
51	X3M	1429.78	17369:47	
52	BJUTThinkingFish	1402.93	24738:49	
53	ElMas7oleen	1402.47	29153:56	
54	BugYou	1398.64	28287:31	
55	PolyuCOMP	1396.64	20278:03	
56	SarajevoDragons	1387.45	10357:47	

					57
EE2CS		1377.34	25520:17		58
ErogeBoys		1375.35	21906:21		
59	asdfgh	1375.00	12903:39		
60	Mafeeesh	1368.82	27884:11		
61	ProjectYasuo	1365.17	24001:22		
62	Team	1362.83	26490:41		
63	50LinesOfCode	1336.89	17412:48		
64	BruteForceBandit...	1334.23	14996:41		
65	NOSPACE	1333.65	26076:40		
66	CapitalControlle...	1318.71	19715:01		
67	METU2	1312.00	9467:39		
68	CUCPpokPOK	1303.28	15826:48		
69	LegacyOfTheVoid	1290.07	28071:42		
70	JUSTAcLimitExcee...	1289.92	15900:30		
71	CPFootLongHamChe...	1289.80	25981:10		
72	DataDons	1288.11	15489:08		
73	SuperNintendoCha...	1284.22	14668:01		
74	ULgTeam5	1264.45	25603:36		
75	sinister	1258.92	26039:25		
76	CodeStudio	1255.16	11213:48		
77	eragon	1254.79	25198:16		
78	XtremeCoderz42	1254.69	15682:08		
79	Friends	1251.48	18788:06		
80	HashPotatoes	1234.51	13591:21		
81	ACDC	1229.69	21078:07		
82	ModerateRedPanda...	1228.62	24332:18		
83	theChorbas	1223.32	13432:48		
84	123Code	1222.85	15741:25		
85	Helli2United	1219.57	13353:25		
86	ccodex	1219.01	27170:22		

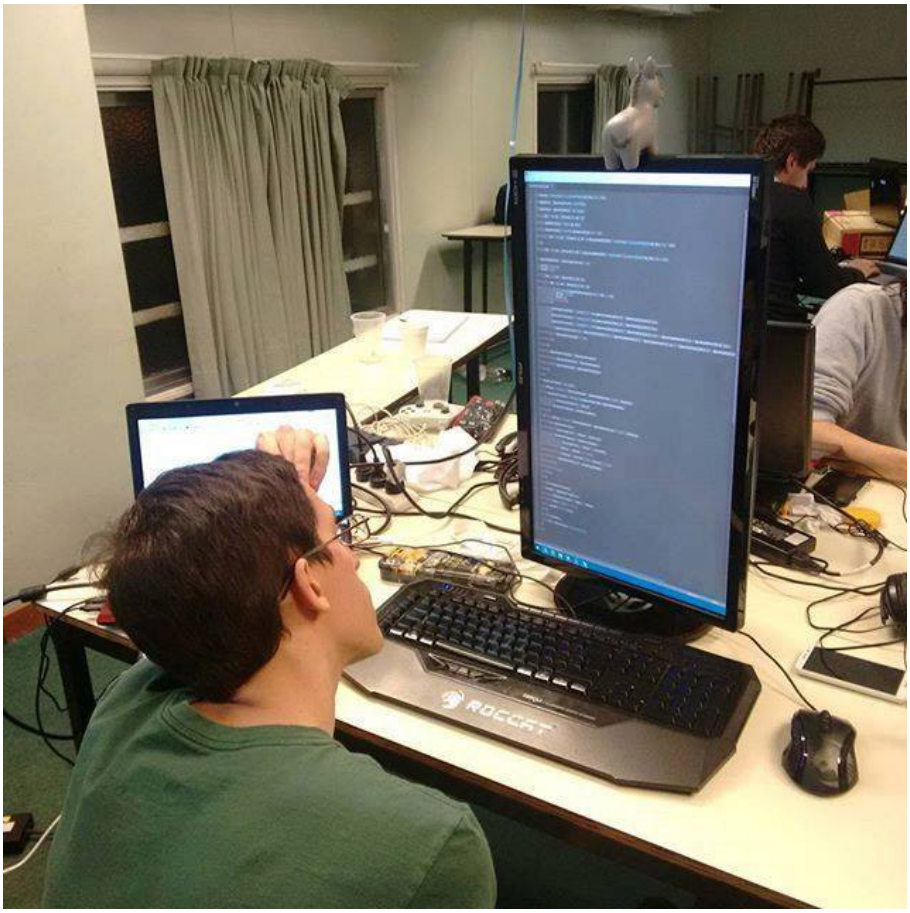
86	hashpi	1219.01	27264:19	
88	AtAvratKod	1208.28	15762:50	
89	NamesAreHard	1205.91	24698:49	
90	TeamGhent	1200.76	14799:10	
91	0xdb	1195.55	27774:17	
92	GoodGuy	1194.46	25055:03	
93	jOmegaTeam	1188.61	27637:11	
94	CUCPNeverSleep	1188.04	28295:45	
95	Uniteam	1187.28	28071:54	
96	MightyMightyPira...	1186.17	19009:56	
97	commandspace	1185.56	27341:19	
98	Binaerbuam	1182.77	27196:43	
99	TheLastGraphBend...	1178.04	21677:20	
100	semicolon2	1173.31	15765:50	

« ‹ 1 2 3 4 5 6 7 8 9 10 › »

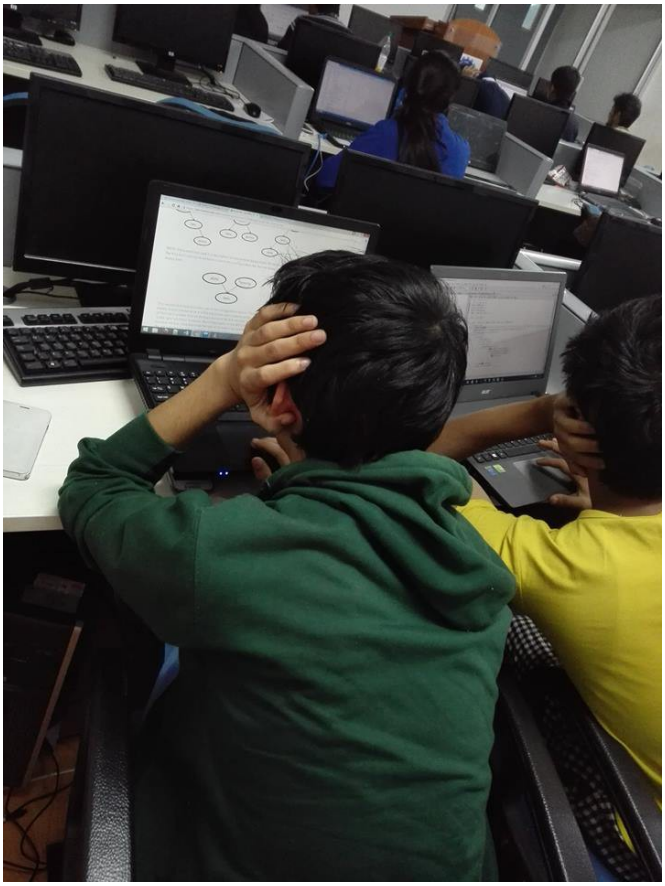
Items per page: 100

Memorable Photos

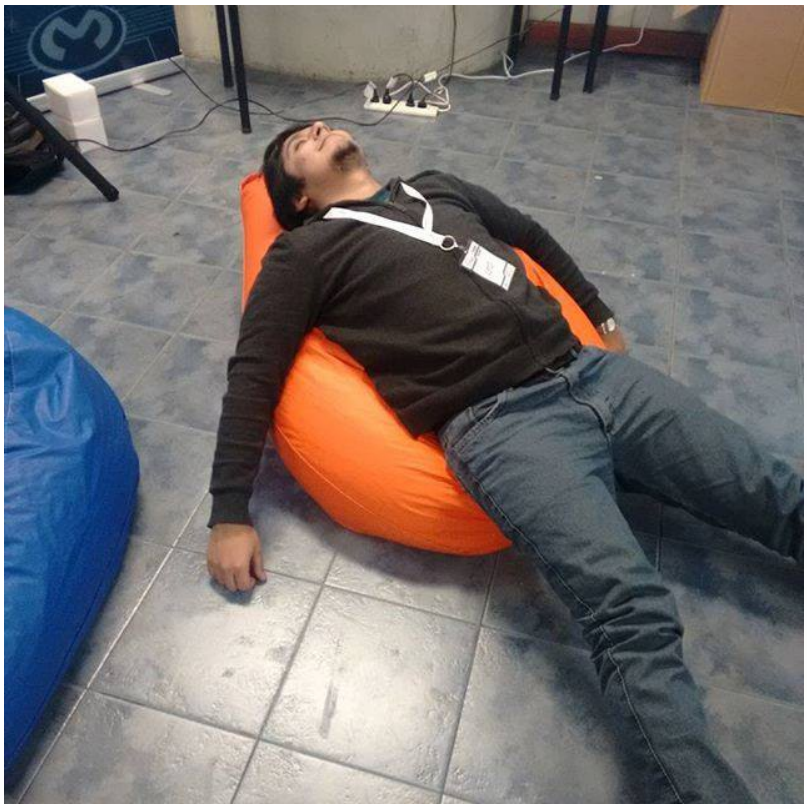

WE CAN CODE IT!


ROCCAT


IEEE
HOT-N-READY




IEEEXtreme9.0 @ IEEE Student branch,University of Peradeniya
©Yasitha Harshanth



CAN'T DECIDE IF I NEED
A HUG, AN XL COFFEE,
6 SHOTS OF VODKA,
OR 2 WEEKS OF SLEEP.

Extras

Practice Makes Perfect!

https://www.hackerrank.com/ieeextreme-challenges

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Please head over to IEEE Extreme contest here. The contest will start at 24 Oct 2015 00:00:00 UTC

IEEExtreme Practice Community

Challenges

Game of Thrones - I

Success Rate: 80.15% Max Score: 100 Difficulty: Easy

Solve Challenge

Magic Square

Success Rate: 83.51% Max Score: 100 Difficulty: Easy

Solve Challenge

Sum it up

Success Rate: 55.56% Max Score: 100 Difficulty: Moderate

Solve Challenge

Back to Square 1

Success Rate: 67.80% Max Score: 100 Difficulty: Moderate

Solve Challenge

Kabloom

Success Rate: 45.45% Max Score: 100 Difficulty: Moderate

Solve Challenge

Play with GCD

Success Rate: 19.05% Max Score: 100 Difficulty: Moderate

Solve Challenge

Current Rank: N/A

In progress. It ends a day from now.

Current Leaderboard

Review Submissions

HackerRank

Demo problem

#1

GAME OF THRONES

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Game of Thrones - I

Problem Statement

Dothraki are planning an attack to usurp King Robert's throne. King Robert learns of this conspiracy from Raven and plans to lock the single door through which the enemy can enter his kingdom.

But, to lock the door he needs a key that is an anagram of a certain palindrome string.

The king has a string composed of lowercase English letters. Help him figure out whether any anagram of the string can be a palindrome or not.

Input Format
A single line which contains the input string.

Constraints
$1 \leq$ *length of string* $\leq 10^5$
Each character of the string is a lowercase English letter.

Output Format
A single line which contains YES or NO in uppercase.

Sample Input : 01

```
aaabbbb
```

Sample Output : 01

```
YES
```

Explanation
A palindrome permutation of the given string is *bbaaabb*.

Sample Input : 02

```
cdefghmnopqrstuvw
```

Sample Output : 02

```
NO
```

Explanation
You can verify that the given string has no palindrome permutation.

Sample Input : 03

```
cdcdcdcdeeeef
```

Sample Output : 03

```
YES
```

Explanation

A palindrome permutation of the given string is *ddcceefeeccdd*.

HackerRank

Magic Square Problem

#2

IEEEXTREME PROGRAMMING COMPETITION 9.0

IEEE

Magic Square

Problem Statement

Magic Square

Johnny designed a magic square (square of numbers with the same sum for all rows, columns and diagonals i.e. both the **main diagonal** - meaning the diagonal that leads from the top-left corner towards bottom-right corner - and the **antidiagonal** - meaning the diagonal that leads from top-right corner towards bottom-left corner). Write a program to test it.

Task
Write a program that will check if the given square is magic (i.e. has the same sum for all rows, columns and diagonals).

Input
First line: **N** , the size of the square (1 <= N <= 600).
Next N lines: The square, N space separated integers pre line, representing the entries per each row of the square.

Output
First line: **M** , the number of lines that **do not sum up** to the sum of the **main diagonal** (i.e. the one that contains the first element of the square). If the Square is magic, the program should output 0.
Next **M** lines: A sorted (in **incremental order**) list of the lines that do not sum up to the sum of the main diagonal. The rows are numbered 1,2,...,N; the columns are numbered -1,-2,...,-N; and the antidiagonal is numbered zero.

Note: There is a newline character at the end of the last line of the output.

Sample Input 1

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

Sample Output 1

```
0
```

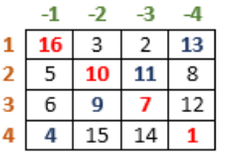
Sample Input 2

```
4
16 3 2 13
5 10 11 8
6 9 7 12
4 15 14 1
```

Sample Output 2

```
3
-2
-1
0
```

Explanation of Sample Output 2
The input square looks as follows:



The square has 4 rows (labeled from 1 to 4 in orange) and 4 columns (labeled from -1 to -4 in green) as depicted in the image above. The main diagonal and antidiagonal of the square are highlighted in red and blue respectively.

The main diagonal has sum = 16 + 10 + 7 +1 = 34.
The antidiagonal has sum = 13 + 11 + 9 + 4 = 37. This is different to the sum of the main diagonal so value 0 corresponding to the antidiagonal should be reported.
Row 1 has sum = 16 + 3 + 2 + 13 = 34.
Row 2 has sum = 5 + 10 + 11 + 8 = 34.
Row 3 has sum = 6 + 9 + 7 + 12 = 34.
Row 4 has sum = 4 + 15 + 14 + 1 = 34.
Column -1 has sum = 16 + 5 + 6 + 4 = 31. This is different to the sum of the main diagonal so value -1 should be reported.
Column -2 has sum = 3 + 10 + 9 + 15 = 37. This is different to the sum of the main diagonal so value -2 should be reported.
Column -3 has sum = 2 + 11 + 7 + 14 = 34.
Column -4 has sum = 13 + 8 + 12 + 1 = 34.
Based on the above, there are 3 lines that do not sum up to the sum of the elements of the main diagonal. Since they should be sorted in incremental order, the output should be:
3
-2
-1
0

Sum it up

HackerRank

Problem Statement

Minka is very smart kid who recently started learning computer programming.
His coach gave him a cyclic array A having N numbers, and he has to perform Q operations on this array.
In each operation the coach would provide him with a number X. After each operation, every element of the cyclic array would be replaced by the sum of itself and the element lying X positions behind it in the cyclic array. All these replacements take place simultaneously.
For example, if the cyclic array was [a, b, c, d], then after the operation with X = 1, the new array would be [a+d, b+a, c+b, d+c].
He needs to output the sum of the elements of the final array modulus 10^9+7.
He made a program for it but it's not very efficient. You know he is a beginner, so he wants you to make an efficient program for this task because he doesn't want to disappoint his coach.

Input

The first line of each test file contains a integer N (1 <= N <= 100000).
The next line contains N space separated integers which represent the elements of the cyclic array (1 <= Ai <= 10^9).
The third line contains a integer Q (0 <= Q <= 1000000) representing the number of operations that will be applied to the array.
Finally, Q lines follow, each one containing an integer X (0 <= X < N).

Output

Your program should output to the standard output stream the **sum of the elements of the final array modulus 10^9+7.**
Note: There is a newline character at the end of the last line of the output.

Sample Input 1

```
5
1 2 3 4 5
2
1
0
```

Sample Output 1

```
60
```

Explanation of Sample Input 1

- After the 1st operation (X = 1), the array would be [1+5, 2+1, 3+2, 4+3, 5+4] = [6, 3, 5, 7, 9]
- After 2nd operation (X = 0), the array would be [6+6, 3+3, 5+5, 7+7, 9+9] = [12, 6, 10, 14, 18]
- Thus the correct answer would equal to = (12+6+10+14+18) % (10^9+7) = 60

Sample Input 2

```
5
```

```
1 2 3 4 5
0
```

Sample Output 2

```
15
```

Back to Square 1

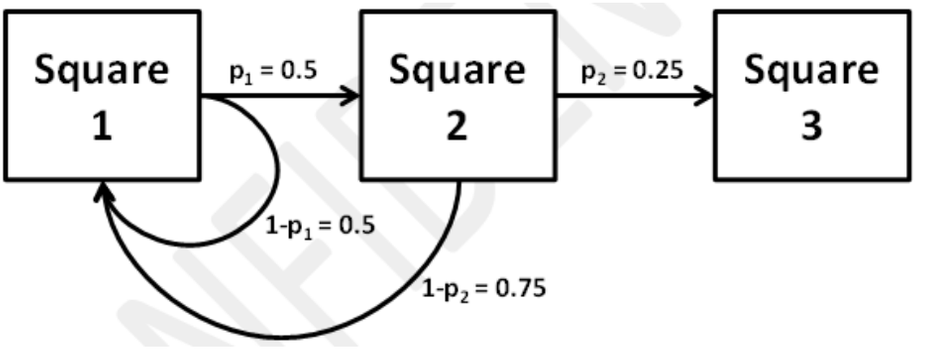
Problem Statement

Back to Square 1

The game “Back to Square 1” is played on a board that has *n* squares in a row and *n-1* probabilities. Players take turns playing. On their first turn, a player advances to square 1.After the first turn, if a player is on square *i* , the player advances to square *i* + 1 with probability *p(i)* , and returns to square 1 with probability 1-*p(i)* .The player is finished upon reaching square *n* .

Task

Write a program that determines the expected number of turns needed for a player to reach the final square.For example, consider the board below with *n* = 3 and *p(1)* = 0.5 and *p(2)* = 0.25. A player moves to square 1 on their first turn. With probability *p(1)* , they move to square 2 on their second turn, but with probability 1- *p(1)* , they remain on square 1. If they were lucky and made it to square 2 on their second turn, they advance to square 3 on their third turn with probability *p(2)* , but they would go back to square 1 with probability 1- *p(2)* . Thus, a really lucky player could finish is 3 turns. However, on average, it would take 13 turns for a player to make it to square 3.



Input

The input is made up of multiple test cases. Each test case contains 2 lines of input.
The first line in each test case is an integer *n* , 1 <= *n* <= 1,000, which represents the number of squares for this test case.
On the next line are *n* -1 single-space separated floating point numbers, each greater than 0 and less than or equal to 1, representing *p(1)* , *p(2)* , *p(3)* , ..., *p(n-1)* , respectively.
The input will end with a 0 on a line by itself.
Note: If for an input test case n=1 (i.e. there is only one square) then there will be no following line since there will be no probabilities. For example, the following input:
2
0.5
1
3
0.1 0.2
0
contains in total 3 test cases. The first one having 2 squares with an in-between transition

probability equal to 0.5, the second test case consists of a single square (and thus no transition probabilities are provided) and the last test case consists of 3 squares with respective transition probabilities equal to 0.1 and 0.2 .

Output

For each test case, output the expected number of turns needed to reach the final state, **rounded to the nearest integer**. You are guaranteed that the expected number of turns will be less than or equal to 1,000,000.

Note: Every line of output should end in a newline character .

Sample Input 1

```
3
0.5 0.25
0
```

Sample Output 1

```
13
```

Sample Input 2

```
2
0.5
4
0.3 0.2 0.1
0
```

Sample Output 2

```
3
228
```

Kabloom

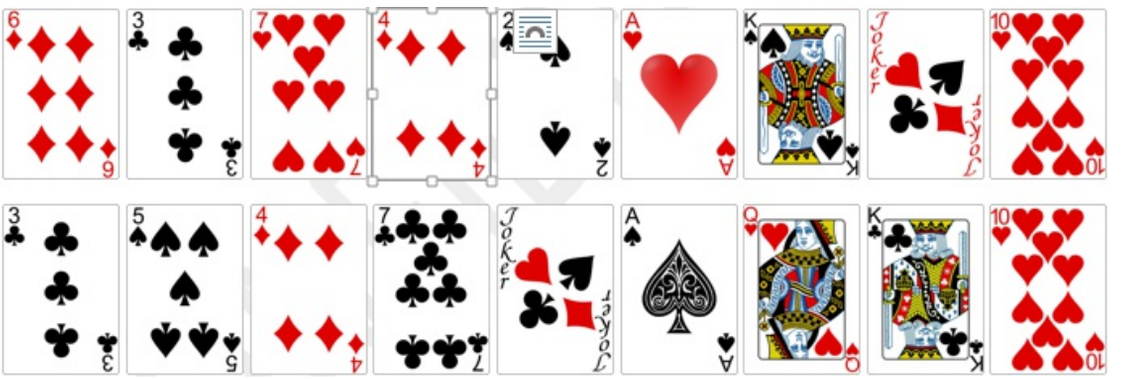
Problem Statement

Kabloom

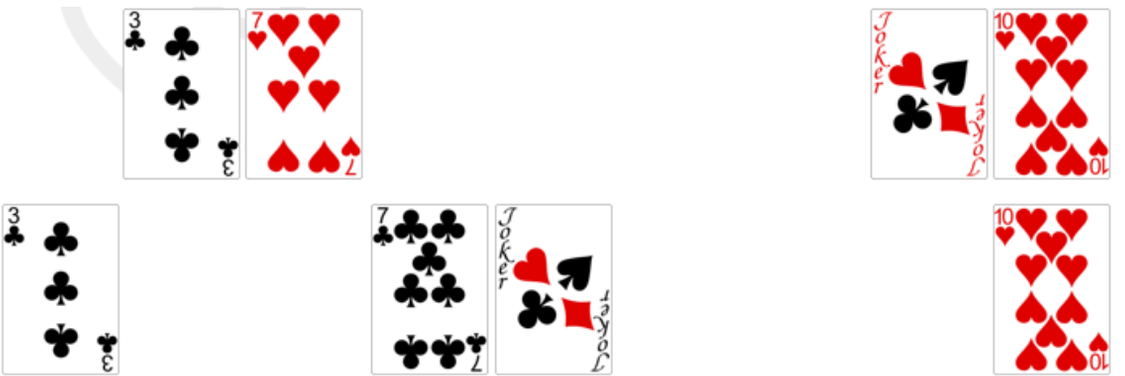
The card game *Kabloom* is played with multiple decks of playing cards. Players are dealt 2 *n* cards, face up and arranged in two rows of *n* cards. The players must discard some of the cards, so that the cards that remain in the first row match the rank of the cards that remain in the second row. The cards match only in rank (e.g. an *Ace* of *Hearts* matches any other *Ace* regardless of suit), but they must appear in the same order in each row. The players are not able to rearrange the order in which the cards appear. Note also that a *Joker* can match any card including another *Joker* .
The goal is to maximize the sum of the point value of the cards that remain. *Aces* are worth 20 points, face cards are worth 15 points, and the numbered cards are worth the number on the card (e.g. the *Seven* of *Clubs* is worth 7 points).The value of a *Joker* is equal to the card with which it is matched, e.g. a *Joker* matched with an *Ace* is worth 20 points, a *Joker* matched with a face card is worth 15 points, etc. If two *Jokers* are matched with each other, they are worth 50 points each.

Task

Write a program that determines the value of the best hand given the two rows of cards. For example, consider the hand that is dealt below.



The best possible hand has a value of 140, and is obtained by keeping the cards shown below:



Card Images by Byron Knoll

Input

The input is made up of multiple test cases (#test cases<=30, if 1<=n<=10 or #test cases<=10 if

10<=n<=1000). Each test case contains three lines of input.
The first line in each test case is an integer n , 1 <= n <= 1,000, indicating how many cards are in each row.
The second line of the test case will contain n symbols representing the ranks of the cards in the first row. Each symbol will be chosen from the list {A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, R}. The symbols in the list represent the following ranks, respectively, {Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King, Joker}. Similarly, the third line of the test case will contain the n symbols of the cards in the second row.
The input will end with a 0 on a line by itself.

Output

For each test case, output the value of the best Kabloom hand on a line by itself. Note that the cards that comprise the best Kabloom hand may not be unique for a test case.
Note: Every line of output should end in a newline character .

Sample Input 1

```
9
6 3 7 4 2 A K R T
3 5 4 7 R A Q K T
0
```

Sample Output 1

```
140
```

Sample Input 2

```
7
R R 5 4 A T Q
Q 3 T A 8 8 8
13
A 2 3 4 5 6 7 8 9 T J Q K
K Q J T 9 8 7 6 5 4 3 2 A
6
A A A A A A
K Q J T 9 8
13
A 2 3 4 5 6 7 8 9 T J Q K
A 2 3 4 5 6 7 8 9 T J Q K
0
```

Sample Output 2

```
90
40
```

Play with GCD

Problem Statement

Play with GCD

Task

Minka is very smart kid who recently started learning computer programming.
He learned how to calculate the Greatest Common Divisor (GCD) of given numbers. The GCD http://en.wikipedia.org/wiki/Greatest_common_divisor of k numbers say [n1,n2,n3... nk] is the largest positive integer that divides all these numbers without any remainder. You may find out more about the GCD and the way it is calculated on the Wikipedia website.
Minka has N (1 <= N <= 10^5) balls and there is a number V (1 <= V <= 10^4) written on every ball.
Now Minka has to perform Q queries, and in each query he wants to know the number of possible ways he can choose balls out of the N balls, so that GCD of the numbers written on the chosen balls equals to the number X of each query. Although he already knows the answer for each query, he would still like you to check if you can also find answer to his queries.
Since number of ways can be very large, your program should output the number of ways modulus 10^9+7.
Notes:
1) There can be at most 100 distinct numbers written on N balls.
2) By definition, the GCD is only defined for 2 or more numbers. For this problem, however, we will consider that the GCD of a single number may also defined and in such case the GCD of a single number will be equal to the number itself (i.e. the GCD of 2 is 2. Please refer to the explanation of Sample Input 1 for more details).

Input

The first line of each test file contains an integer N (1 <= N <= 10^5) denoting the number of balls.
The next line contains N space separated integer numbers, each one representing the number written on each of the N balls. The ith number (Vi) corresponds to the number written on the ith ball (1 <= Vi <= 10^4).
The third line contains an integer Q (1 <= Q <= 10^4) representing the number of GCD queries that will have to be performed.
Finally, Q lines follow, each one containing an integer X (1 <= X <= 10^4) corresponding to the GCD of each query.

Output

Your program should output the number of ways modulus 10^9+7 that balls can be drawn from the set, so that their GCD equals the number X corresponding to each query.
Note: There is a newline character at the end of the last line of the output.

Sample Input 1

```
5
2 3 5 6 6
2
2
5
```

Sample Output 1

```
4
1
```

Explanation of Sample Input 1

We have 5 balls in the set, labeled with numbers [2, 3, 5, 6, 6] respectively. For the first query (X=2), there are in total 4 (distinct) ways by which we may choose balls so that their GCD equals 2, meaning:
a) {1, 4} (i.e. ball 1 labeled with number 2 and ball 4 labeled with number 6)
b) {1, 5} (i.e. ball 1 labeled with number 2 and ball 5 labeled with number 6)
c) {1, 4, 5} (i.e. ball 1 labeled with number 2, ball 4 labeled with number 6 and ball 5 labeled with number 6)
d) {1} (i.e. ball 1 labeled with number 2 since according to our definition of GCD, the GCD of 2 would equal 2)

Regarding the second query (X=5), there is only one way to choose balls so that their GCD equals 5, which is to choose only ball 3 (labeled with number 5).

IEEE Xtreme Past Problems

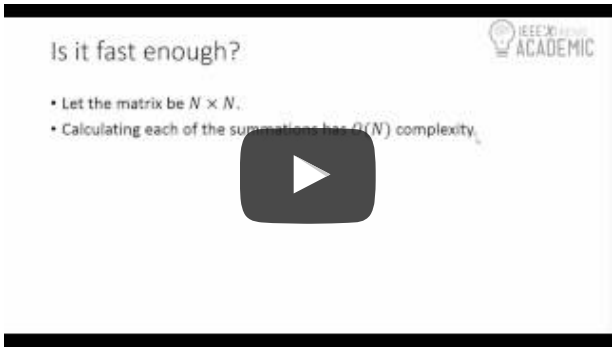
Jeremy Blum, Shervin Shahidi / IEEE Xtreme

Solving past IEEE Xtreme problems.

1. Back to Square I (/docs/24b)

This video explains an approach to solving the "Back to Square 1" problem from the IEEEXtreme8 competition. Before viewing the

Go to module (/docs/24b)



2. Magic Square (/docs/54b)

Go to module (/docs/54b)

Contact Us

(http://facebook.com/IEEEAcademic)

(http://twitter.com/IEEEAcademic)

(http://youtube.com/IEEEAcademic)

Get in touch!
(/contactus)

About Us

How it began (/about#story)
What is IEEE Academic (/about#academic)
What is IEEE (/about#IEEE)
Meet the Team (/about#participants)

More Links

Information flyer (/flyer-2014-05.pdf)
Pitch slide deck (/IEEEAcademic-pitchdeck.pdf)
Press release

Privacy & Cookies (http://www.ieee.org/security_privacy.html)
Terms and Conditions (http://www.ieee.org/site_terms_conditions.html)

* Why Beta? (/whybeta)

#IEEEXtreme

</html>

The **end** is always near.