(Lab 3) Problem Set 2:
Control Structures

P2 Solutions limited in scope to:		

• P1 Concepts	• Selection statements
	  ○ single selection
	  ○ double selection
	  ○ multi-selection
		• Repetition statements
		  ○ counter-controlled loops
		  ○ sentinel-controlled loops

Submission Rules:

1. Submissions must be zipped into a **handin.zip** file. Each problem must be implemented in its own class file. Use the name of the problem as the class name.

2. You must use standard input and standard output for ALL your problems. It means that the input should be entered from the keyboard while the output will be displayed on the screen.

3. Your source code files should include a comment at the beginning including your name and that problem number/name.

4. The output of your solutions must be formatted exactly as the sample output to receive full credit for that submission.

5. Compile & test your solutions before submitting.

6. Each problem is worth up to 10 points total. The breakdown is as follows: 2 points for compiling, 3 points for correct output with sample inputs, 5 points for additional inputs.

7. This lab is worth a max total of: 40 points. You can complete as many problems as you like, but cannot receive more than 40 points towards the lab grade. All points in excess of that are for bragging rights. (Check the scoreboard to see how you did!)

8. Submission:

- You have unlimited submission attempts until the deadline passes
- You'll receive your lab grade immediately after submitting
- **IMPORTANT:** if your grade is lower than 70% when the deadline passes, then you must attend a recitation session & get TA signoff to receive full credit for that lab challenge.
- **Online help/discussions**: www.acmuno.slack.com, **channel: *#1583-java1***

Problem 1: Prime Number (10 points)

(Cyber Security) A number is considered prime if its only divisible by itself and 1. Note that 1 is not prime. Prime numbers are commonly used in many current encryption techniques. So it's very important to be able to identify whether a number is prime or not. Write a program that if given an integer number, will determine if it is prime.

Facts

- You must check the given number to every possible smaller number
- If no smaller factor divides wholly then that number must be prime
- if a smaller factor divides wholly then that number is not prime

Input

The first line will be the number of test cases. Each line afterwards, is a single integer input.

Output

Your program must print boolean result whether the number is prime or not

Sample Input	**Sample Output**
4 2 17 30 131101	true true false true

Problem 2: RSA Public Key (10 points)

(Cyber Security) RSA is an asymmetric encryption scheme, where a public key is used to encrypt data and a different, private key decrypts that data. RSA public/private keys are generated from two prime numbers, typically very large ones. The idea behind RSA is based on the fact that its difficult to factorize very large integers. RSA public key generation is done in part by using the composite number (*n*) between two prime numbers (*p,q*), and some other coprime number (*e*) which is not a divisor of *n*. Determine if two seed numbers and the given *e* are valid for generating a RSA key, and if so, calculate *n*, the basis for generating private/public keys.

Facts

RSA Public/Private key generation requires:

- *p*, one of the given numbers, it must be prime.
- *q*, one of the given numbers, it must be prime.
- *n* is a composite number where *p * q*
- *totient(n) = (p - 1) * (q - 1)*
- *e* is a coprime to n (i.e. not divisible), and must be between 1 < *totient(n)*.
- Consider using `long` data types instead of `int`

Input

First line represents the number of test cases. Each line there after, consists of three positive integers. The first integer represents the *p* terms, the second integer represents the *q* term, the third integer represents the *e* term.

Output

For each pair of integers, determine if they are not valid for key generation. if not, either print "Invalid n for RSA Key!" or "Invalid e for RSA Key!" depending on which is not valid. Otherwise, print "RSA Public Key: n=%d e=%d" with n and e respectively.

Sample Input	**Sample Output**
4 5 3 7 33 7 5 131101 524269 17 5 3 17	RSA Public Key: n=15 e=7 Invalid n for RSA Key! RSA Public Key: n=68732190169 e=17 Invalid e for RSA Key!

Problem 3: RSA Private Key (10 points)

(Cyber Security) In the RSA algorithm, encrypting and decrypting a message is done using a pair of numbers that are multiplicative inverses with respect to a carefully selected modulus. In this task, you must calculate the modular multiplicative inverse for given set of values.

What is the Modular Multiplicative Inverse? In mathematics, each operation typically has an inverse. For example, the inverse of multiplication is division and the inverse of addition is subtraction. Likewise, modulus has an inverse expression. The basic expression given is:

Solve for ***x,*** Given an integer ***a*** and modulo ***m*** where ***ax % m = 1***

.

However, solving for x is not a trivial task. One naive approach is to bruteforce a solution for x by simply trying each number starting at 1 until a solution is found that makes the expression `true`.

Facts

- Solve for **x**, when ***(a*x) % m == 1*** and where ***a*** and **m** are given.
- *Use a Brute force approach:* First try 1 for x, and check for validity. if invalid then continue to increase x until you find a valid value for x that makes the expression true.
- Consider using `long` data types instead of `int`

Input

The first line of input is the number of test cases. Each line thereafter, consists of two positive long integer inputs. The first represents the integer a, the second represents the modulo m.

Output

Your program should print the value of x that makes the Modular Multiplicative Inverse expression true.

Sample Input	**Sample Output**
3 17 20 3 7 131101 524269	13 5 229125

Problem 4: Player Move Overworld (10 points)

(Game Development) You're the lead programmer for an indie game studio making a retro-style game called Zeldar. You've been tasked to implement the player movement. The game is top-down, with the overworld modeled as a 2d grid. The player's location is tracked by x,y values correlating to its row and column positions within that grid. Given the current position of the player and a sequence of input commands: w,a,s,d you must determine the new position of the player.

Facts

- the player's position is modeled using two integer values (x, y)
- x represents the column position, left-right axis
- y represents the row position, up-down axis
- “w” move up by decreasing y by 1
- “a” move left by decreasing x by 1
- “s” move down by increasing y by 1
- “d” move right by increasing x by 1

Input

The input will begin with a single line containing the number of test cases to execute. The next line should consist of the starting (x,y) position of the player. The next line is the sequence of moves represented with "w", "a", "s", or "d". This sequence can be empty string to any number of letters long. Each input is separated by a space with the last terminated with a new line.

Output

The program should print out the final location of the player in the form of <x> <y>, where “x” and “y” are the coordinates on the overworld grid.

Sample Input	**Sample Output**
2 0 0 w w a a a 9 4 s d w a	-3 -2 9 4

Problem 5: Player Move Dungeon (10 points)

(Game Development) You're the lead programmer at a AAA studio making a sequel to the big hit game, Zeldar 2. You've been challenged to implement player movement in dungeons. The game is top-down, with dungeons modeled as a 2d grid with walls at the edges. The player's location is tracked by x,y values correlating to its row and column positions. Given the current position of the player and a sequence of input commands: w,a,s,d you must determine the new position of the player. The player must not be able to move outside the walls of the dungeon (i.e. grid)

Facts

- the player's position is modeled using two integer values (x, y)
- x represents the column position, left-right axis
- top-left corner is (0,0)
- y represents the row position, up-down axis
- “w” move up by decreasing y by 1
- “a” move left by decreasing x by 1
- “s” move down by increasing y by 1
- “d” move right by increasing x by 1
- if an input attempts to move player off grid, then ignore that move.

Input

The first input is the number of test cases. Each test case contains three lines of inputs. The first line is two positive integers that represent the dungeon's grid size, rows (length) columns (width). The second line is two non-negative integers representing the player's position in the dungeon grid, x,y. The third line represents the sequence of player movements "w", "s", "a", "d".

Output

The program should print the final location of the player in the form of <x> <y>, where “x” and “y” are the coordinates within the dungeon grid.

Sample Input	**Sample Output**
2 4 4 2 3 s s s w 10 10 9 4 s d w a	2 2 8 4

Problem 6: FizzBuzz (10 points)

(General) FizzBuzz is one of the most frequently asked questions on programming interviews and used to filter programmers who can't program. It looks extremely simple but it's tricky for those programmers or coder who struggle to structure their code. Fizzbuzz problem statement is very simple, write a program which return "fizz" if the number is a multiplier of 3, return "buzz" if its multiplier of 5 and return "fizzbuzz" if the number is divisible by both 3 and 5. If the number is not divisible by either 3 or 5 then it should just return the number itself.

Facts

- if number is a multiplier of 3, then print "fizz"
- if number is a multiplier of 5, then print "buzz"
- if number is a multiplier of both 3 and 5, then print "fizzbuzz"
- if number is not a multiplier of either 3 or 5, then print number

Input

First line is the number of test cases. Each line thereafter is an integer.

Output

For each test case, display one of the following text options based from the facts listed: "fizz", "buzz", "fizzbuzz", or the given number

Sample Input	**Sample Output**
3 3 5 15	fizz buzz fizzbuzz

Problem 7: Simple Calculator (10 points)

(General) Calculators represent the most basic, general-purpose of computing machines. Your task is to reduce your highly capable computer down into a simple calculator. You will have to parse a given mathematical expression, and display its result. Your calculator must support addition (+), subtraction (-), multiplication (*), division (/), modulus(%), and exponentiation (**).

Facts

- Mathematical expressions in the form of: ***num1 operator num2***
- Exponentiation should be in the form ***base ** exponent***
    - Example: 3 ** 2 = 9

Input

First line is the number of test cases. Each line thereafter is an integer-based mathematical expression in the form defined in the above facts.

Output

For each test case, display the result of the mathematical expression terminated with a new line character

Sample Input	**Sample Output**
4 3 + 7 5 - 1 4 ** 2 19 / 5	10 4 16 3

Problem 8: Factorials (10 points)

(General) In mathematics, a factorial of a non-negative number n, denoted by n!, is the product of all positive integers less than or equal to n. For example, the value of “3!” is equal to 3*2*1=6. The factorial operation is encountered in many areas of mathematics, notably in algebra and mathematical analysis. Write a program that gives the factorial of integers between 0 to 20 inclusive.

Facts

- The value of 0! is 1.
- Factorials are the multiplicative product of an entire sequence
    - Hence, logic is similar to summing a sequence, but you take the produce instead.

Input

The first line is the number of test cases. Each line thereafter will consist of a integer to be solved as a factorial.

Output

The output must list the solution for the appropriate input value.

Sample Input	**Sample Output**
3 3 2 4	6 2 24

Problem 9: Fibonacci (10 points)

(General) In mathematics, the Fibonacci numbers are the numbers in the following integer sequence, called the Fibonacci sequence, and characterized by the fact that every number after the first two is the sum of the two preceding ones:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Hence, a Fibonacci number *n* may be found with $f(n) = f(n-1) + f(n-2)$.

Write a simple program that finds the the Fibonacci number at a specified position in the Fibonacci series.

Facts

- Each number in Fibonacci is the sum of the previous two numbers in sequence.
- FIbonacci of 0 equals 0, i.e. $f(0) = 0$
- The starting two values in the series for this formula may be considered as 0 and 1
- In order to find a specific Fibonacci number, you must first find all of its preceding fibonacci numbers using the formula above.

Input

The first input is the number of test cases. Each additional input is a non-negative integer and represents the specified index into the Fibonacci sequence.

Output

Print the value at that position in the Fibonacci series.

Sample Input	**Sample Output**
4 1 2 4 20	1 1 3 6765