Practice 0: FizzBuzz

(We know, we know. Let's start with the simple and the known!)

All children play FizzBuzz. I still have fond memories of playing it myself, sitting alone for hours, staring at the wall, whispering the string representations of the integers from 1 up to *n*, for some large value of *n*. For multiples of three, I would whisper "Fizz" instead of the number. For multiples of five, I would whisper "Buzz" instead of the number. And, for those exciting multiples of *both* three and five, I would whisper "FizzBuzz" instead of the number.

I shiver just thinking about it. Ah, the halcyon days of yore ...

You should write a method that will take as input a positive integer *n* and return an array of the FizzBuzz representations of all integers between 1 and that number *n*.

Input/Output Format

Input:

The first line in the test data file contains the number of test cases, and each line after that contains a test case. Each test case consists of a single positive integer **n**.

Output:

For each test case, your program should return a String[] array of the FizzBuzz string representations of each integer between 1 and **n**.

Note:

We have provided a skeleton program that reads the input and prints the output based on the `doFizzBuzz(int n)` method that you will implement.

Examples:

Input:	**Output:**
2 3 15	1 2 Fizz 1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz

Practice 1: Esports Scoring

You are in charge of recording the points won by a particular esports team, the UMD Electronic Terrapins, over multiple rounds of a game. The scoring rules for that game are a bit odd, and one round's score can impact the scores of future rounds. Information about scores for each round are reported as one of four strings:

- Integer (one round's score): Directly represents the number of points the team received in a round. E.g., the string "4" represents 4 points.
- "+" (one round's score): The points received in this round are the sum of the last two *valid* rounds' scores. E.g., if the last two valid rounds had points 3 and 5, then the string "+" represents 3+5 = 8 points for a particular round.
- "D" (one round's score): The points received in this round are double the points received in the last *valid* round. E.g., if the last valid round had 4 points, then the string "D" represents 2*4 = 8 points for a particular round.
- "R" (an operation, which is not a round's score): Remove the last valid round's points from the total score. For example, if the last valid round had 4 points recorded, then the string "R" would delete those 4 points for the cumulative score.

Each round's operation is permanent and could have an impact on the round before and the round after.

You must write a method that takes as input a String[] array of operations and returns the cumulative score of that team across all rounds.

Input/Output Format

Input:

The first line in the test data file contains the number of test cases, and each line after that contains a test case. Each test case consists of a single integer **n** followed by **n** Strings (either integers, "+", "D", or "R") representing the team's scoring over **n** rounds of play.

- The size of the input list will be between 1 and 10,000.
- Every integer represented in the list will be between -50,000 and 50,000.

Output:

For each test case, your program should output the cumulative score, an integer.

Note:

We have provided a skeleton program that reads the input and prints the output based on the `countPoints(int len, String[] ops)` method, which you will implement.

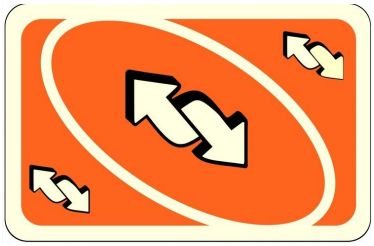
Examples:

Input:	**Output:**
4 3 1 1 1 3 1 1 + 5 5 2 R D + 8 5 -2 4 R D 9 + +	3 4 30 27

An explanation of one of the examples, 5 2 R D +, follows:
Round 1: Record 5 points. The current total is: 5.
Round 2: Record 2 points. The current total is: 7.
Operation 1: The previous valid round, Round 2, was invalid. Remove its 2 points. Current total: 5.
Round 3: The previous valid round, Round 1 (because Round 2 was removed), gives you double points. Record 2*5=10 points for this round. The current total is: 15.
Round 4: Add up the last two valid rounds' points: 5 + 10 = 15. Record 15 points for this round. The current total is: 30.

Practice 2: Swap It to the Max

Given a nonnegative integer, you are allowed to swap at most two digits. Write a method that takes as input a nonnegative integer and returns the maximum integer that could result from swapping up to two digits. For example, if the input is 12345, then your method would return 52341, the maximum integer available after swapping two digits (in this case, swapping the “5” and the “1”).



Swap it!

Input/Output Format

Input:

The first line in the test data file contains the number of test cases, and then the test cases are listed one by one. Each test case is a single nonnegative integer **n**. You can assume **n** is at most 10^8, i.e., it is well below 2^{31}.

Output:

For each test case, your method should return the maximum integer that can be constructed after swapping at most two digits in **n**.

Examples:

Input:	**Output:**
3 12345 99999 112233	52341 99999 312231

Practice 3: Valid Subsequence of Parentheses

Given a non-empty string containing only the characters '(' and ')', write a method that computes and returns the length of the longest valid (well-formed) parentheses substring.

Input/Output Format

Input:

The first line in the test data file contains the number of test cases, and then the test cases are listed one by one. Each test case is simply a non-empty string consisting solely of '(' and ')' characters, e.g., "()())((".

Output:

For each test case, your program should output an integer representing the length of the longest valid substring.

Note:

We have provided a skeleton program that reads the input and prints the output based on the following `longestValid()` method. It should return an integer representing the longest valid substring of parentheses in the input String parens.

```
private static int longestValid(String parens)
```

Examples:

Input:	**Output:**
3 () ((()) (((()))	2 4 6