{1} Valid Perfect Square

Given a positive integer num, return true if num is a perfect square or false otherwise.

A perfect square is an integer that is the square of an integer. In other words, it is the product of some integer with itself.

You must not use any built-in library function, such as sqrt.

Example 1:

Input: num = 16

Output: true

Explanation: We return true because 4 \* 4 = 16 and 4 is an integer.

Example 2:

Input: num = 14

Output: false

Explanation: We return false because 3.742 \* 3.742 = 14 and 3.742 is not an integer.

class Solution {

public boolean isPerfectSquare(int num) {

long s=1,e=num;

while(s<=e)

{

long mid=s+(e-s)/2;

if(mid\*mid==num)

return true;

else if(mid\*mid>num)

e=mid-1;

else

s=mid+1;

}

return false;

}

}

{2} Convert a Number to Hexadecimal

Given an integer num, return a string representing its hexadecimal representation. For negative integers, two’s complement method is used.

All the letters in the answer string should be lowercase characters, and there should not be any leading zeros in the answer except for the zero itself.

Note: You are not allowed to use any built-in library method to directly solve this problem.

Example 1:

Input: num = 26

Output: "1a"

Example 2:

Input: num = -1

Output: "ffffffff"

class Solution {

public String toHex(int num) {

if(num==0) return "0";

String s = "";

String hex[] = {"0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f"};

for(int i=1;i<=8;i++){

int modified\_num = num & 15;

s = hex[modified\_num]+s;

num = num>>4;

if(num==0) break;

}

return s;

}

}

{3}Fizz Buzz

Given an integer n, return a string array answer (1-indexed) where:

answer[i] == "FizzBuzz" if i is divisible by 3 and 5.

answer[i] == "Fizz" if i is divisible by 3.

answer[i] == "Buzz" if i is divisible by 5.

answer[i] == i (as a string) if none of the above conditions are true.

Example 1:

Input: n = 3

Output: ["1","2","Fizz"]

Example 2:

Input: n = 5

Output: ["1","2","Fizz","4","Buzz"]

Example 3:

Input: n = 15

Output: ["1","2","Fizz","4","Buzz","Fizz","7","8","Fizz","Buzz","11","Fizz","13","14","FizzBuzz"]

class Solution {

public List<String> fizzBuzz(int n) {

List<String> ans = new ArrayList<>();

for (int i = 1; i <= n; i++) {

if (i % 15 == 0) {

ans.add("FizzBuzz");

} else if (i % 3 == 0) {

ans.add("Fizz");

} else if (i % 5 == 0) {

ans.add("Buzz");

} else {

ans.add(String.valueOf(i));

}

}

return ans;

}

}

{4} Add Strings

Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string.

You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.

Example 1:

Input: num1 = "11", num2 = "123"

Output: "134"

Example 2:

Input: num1 = "456", num2 = "77"

Output: "533"

Example 3:

Input: num1 = "0", num2 = "0"

Output: "0"

class Solution {

public String addStrings(String num1, String num2) {

String sb = new String();

int i = num1.length()-1;

int j = num2.length()-1;

int cry = 0;

while(i >= 0 || j >= 0)

{

if(i >= 0)

cry += (int)(num1.charAt(i--) - '0');

if(j >= 0)

cry += (int)(num2.charAt(j--) - '0');

int digit = cry%10;

sb = digit + "" + sb;

cry/= 10;

}

return (cry == 0)? sb : cry + "" + sb;

}

}

{5} Arranging Coins

You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase may be incomplete.

Given the integer n, return the number of complete rows of the staircase you will build.

Example 1:

Input: n = 5

Output: 2

Explanation: Because the 3rd row is incomplete, we return 2.

Example 2:

Input: n = 8

Output: 3

Explanation: Because the 4th row is incomplete, we return 3.

class Solution {

public int arrangeCoins(int n) {

int ans = 1;

while(n > 0){

ans++;

n = n-ans;

}

return ans-1;

}

}

{6}Construct the Rectangle

A web developer needs to know how to design a web page's size. So, given a specific rectangular web page’s area, your job by now is to design a rectangular web page, whose length L and width W satisfy the following requirements:

The area of the rectangular web page you designed must equal to the given target area.

The width W should not be larger than the length L, which means L >= W.

The difference between length L and width W should be as small as possible.

Return an array [L, W] where L and W are the length and width of the web page you designed in sequence.

Example 1:

Input: area = 4

Output: [2,2]

Explanation: The target area is 4, and all the possible ways to construct it are [1,4], [2,2], [4,1].

But according to requirement 2, [1,4] is illegal; according to requirement 3, [4,1] is not optimal compared to [2,2]. So the length L is 2, and the width W is 2.

Example 2:

Input: area = 37

Output: [37,1]

Example 3:

Input: area = 122122

Output: [427,286]

class Solution {

public int[] constructRectangle(int area) {

int width=(int)Math.sqrt(area);

while(area%width>0)

--width;

return new int[]{area/width,width};

}

}

{7}Base 7

Given an integer num, return a string of its base 7 representation.

Example 1:

Input: num = 100

Output: "202"

Example 2:

Input: num = -7

Output: "-10"

class Solution {

public String convertToBase7(int num) {

int count=0,res=0;

while(num!=0)

{

int dig=num%7;

res=res+dig\*(int)Math.pow(10,count);

count++;

num=num/7;

}

return Integer.toString(res);

}

}

{8}Perfect Number

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. A divisor of an integer x is an integer that can divide x evenly.

Given an integer n, return true if n is a perfect number, otherwise return false.

Example 1:

Input: num = 28

Output: true

Explanation: 28 = 1 + 2 + 4 + 7 + 14

1, 2, 4, 7, and 14 are all divisors of 28.

Example 2:

Input: num = 7

Output: false

class Solution {

public boolean checkPerfectNumber(int num) {

if (num == 1)

{

return false;

}

int sum = 0;

for (int i = 2; i <= Math.sqrt(num); i++)

{

if (num % i == 0) {

sum =sum+i;

if (i != num / i) sum += num / i;

}

}

sum++;

return sum == num;

}

}

{9}Fibonacci Number

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

F(0) = 0, F(1) = 1

F(n) = F(n - 1) + F(n - 2), for n > 1.

Given n, calculate F(n).

Example 1:

Input: n = 2

Output: 1

Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.

Example 2:

Input: n = 3

Output: 2

Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.

Example 3:

Input: n = 4

Output: 3

Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.

class Solution {

public int fib(int n) {

if(n<= 1)

return n;

int a = 0, b = 1;

while(n-- > 1)

{

int sum = a + b;

a = b;

b = sum;

}

return b;

}

}

{10}Range Addition II

You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.

Count and return the number of maximum integers in the matrix after performing all the operations.

Example 1:

Input: m = 3, n = 3, ops = [[2,2],[3,3]]

Output: 4

Explanation: The maximum integer in M is 2, and there are four of it in M. So return 4.

Example 2:

Input: m = 3, n = 3, ops = [[2,2],[3,3],[3,3],[3,3],[2,2],[3,3],[3,3],[3,3],[2,2],[3,3],[3,3],[3,3]]

Output: 4

Example 3:

Input: m = 3, n = 3, ops = []

Output: 9

class Solution {

public int maxCount(int m, int n, int[][] ops) {

int min\_row=m;

int min\_col=n;

for(int[]op:ops)

{

min\_row=Math.min(min\_row,op[0]);

min\_col=Math.min(min\_col,op[1]);

}

return min\_row\*min\_col;

}

}