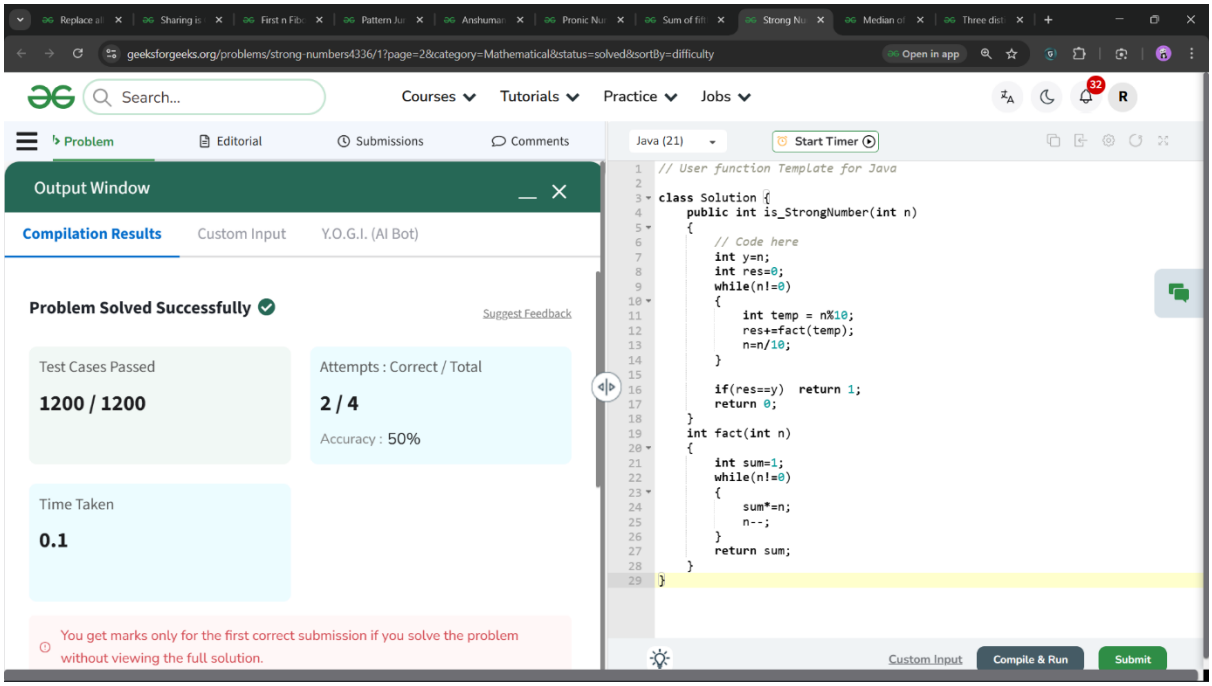
PROGRAMS ON MATH LOGIC

1 . Strong Numbers :

Strong Numbers are the numbers whose sum of factorial of digits is equal to the original number. Given a number, the task is to check if it is a Strong Number or not.

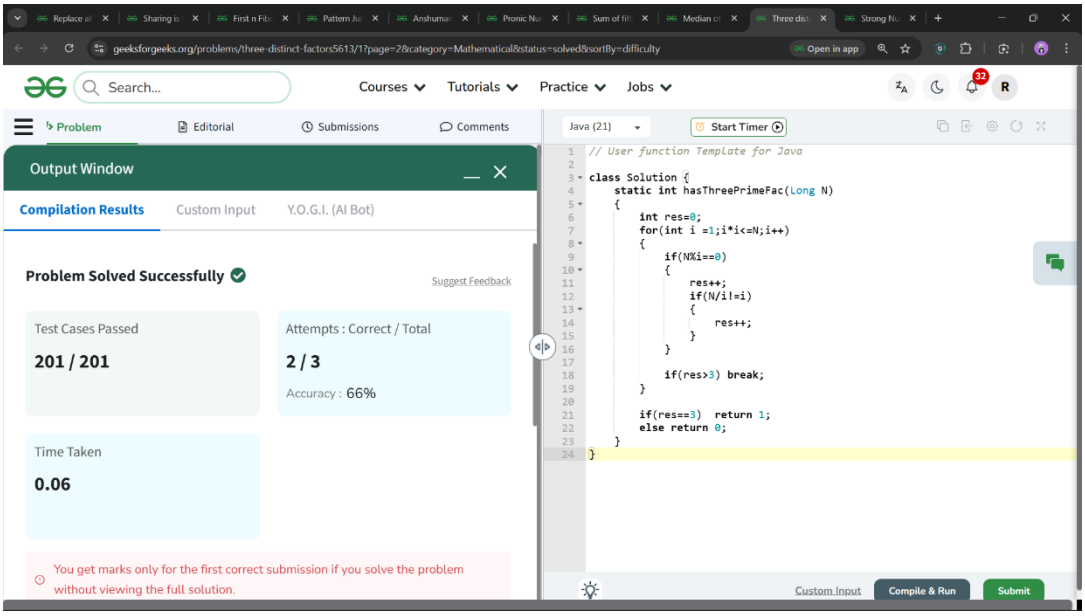
```
class Solution {
public int is_StrongNumber(int n)
{
    int y=n;
    int res=0;
    while(n!=0)
    {
        int temp = n%10;
        res+=fact(temp);
        n=n/10;
    }
    if(res==y)  return 1;
    return 0;
}
int fact(int n)
{
    int sum=1;
    while(n!=0)
    {
        sum*=n;
        n--;
    }
    return sum;
}
}
```


geeksforgeeks.org/problems/strong-numbers4336/1?page=2&category=Mathematical&status=solved&sortBy=difficulty
Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1200 / 1200
Attempts : Correct / Total
2 / 4
Accuracy : 50%
Time Taken
0.1
You get marks only for the first correct submission if you solve the problem without viewing the full solution.
Java (21)
Start Timer
// User function Template for Java
class Solution {
public int is_StrongNumber(int n)
{
// Code here
int y=n;
int res=0;
while(n!=0)
{
int temp = n%10;
res+=fact(temp);
n=n/10;
}
if(res==y) return 1;
return 0;
}
int fact(int n)
{
int sum=1;
while(n!=0)
{
sum*=n;
n--;
}
return sum;
}
}
Custom Input
Compile & Run
Submit

2 . Three distinct factors

Given a positive integer **N**. The task is to check whether a number has exactly three distinct factors or not.

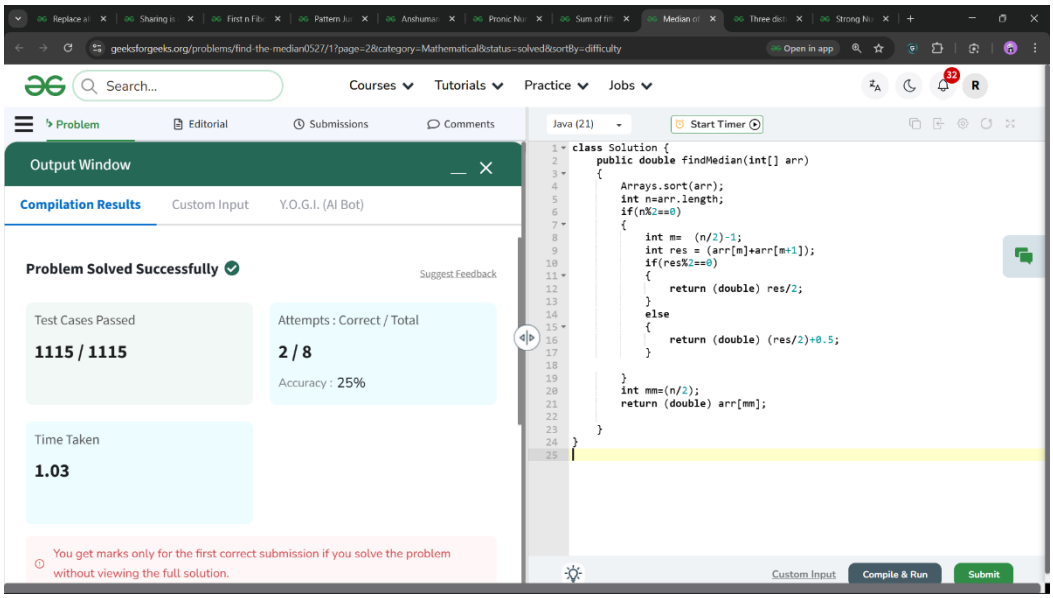
```
class Solution {
   static int hasThreePrimeFac(Long N)
   {
      int res=0;
      for(int i =1;i*i<=N;i++)
      {
         if(N%i==0)
         {
            res++;
            if(N/i!=i)
            {
               res++;
            }
         }

         if(res>3) break;
      }

      if(res==3)  return 1;
      else return 0;
   }
}
```



3 .Median of an Array

Given an array arr[] of integers, calculate the median.

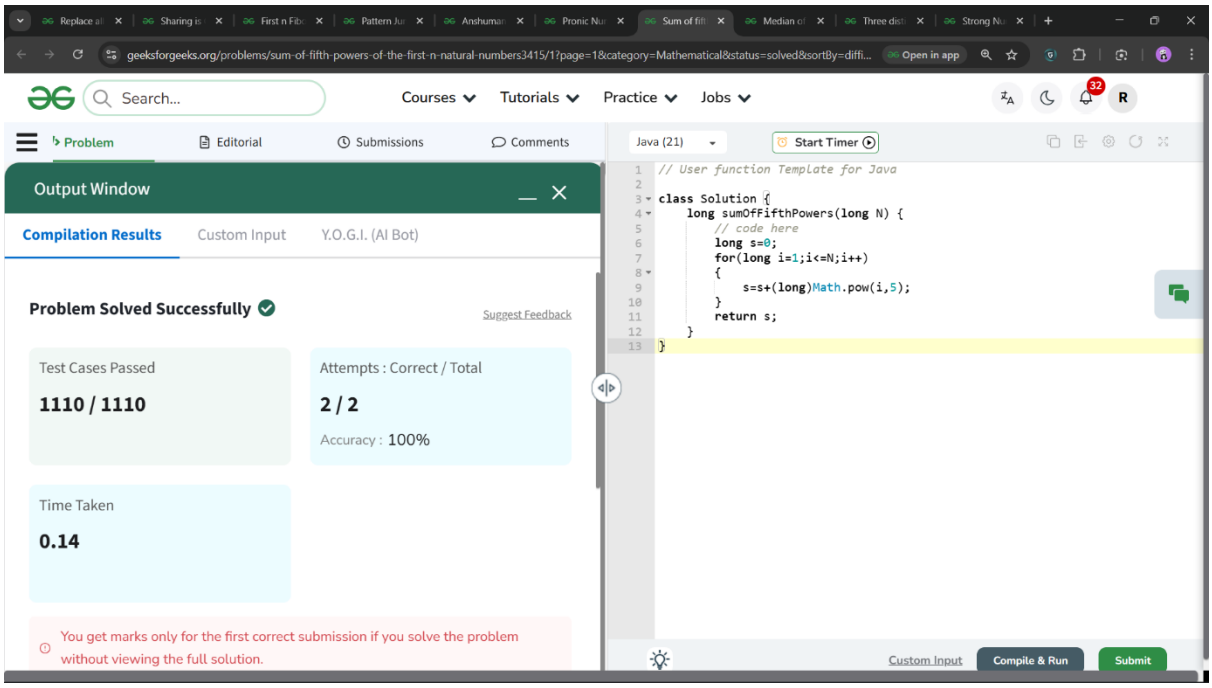
```
class Solution {
   public double findMedian(int[] arr)
   {
      Arrays.sort(arr);
      int n=arr.length;
      if(n%2==0)
      {
         int m=  (n/2)-1;
         int res = (arr[m]+arr[m+1]);
         if(res%2==0)
         {
            return (double) res/2;
         }
         else
         {
            return (double) (res/2)+0.5;
         }

      }
      int mm=(n/2);
      return (double) arr[mm];

   }
}
```



4 . Sum of fifth powers of the first n natural numbers

Given a number N.Find the sum of fifth powers of natural numbers till N i.e. $1^5+2^5+3^5+..+N^5$.

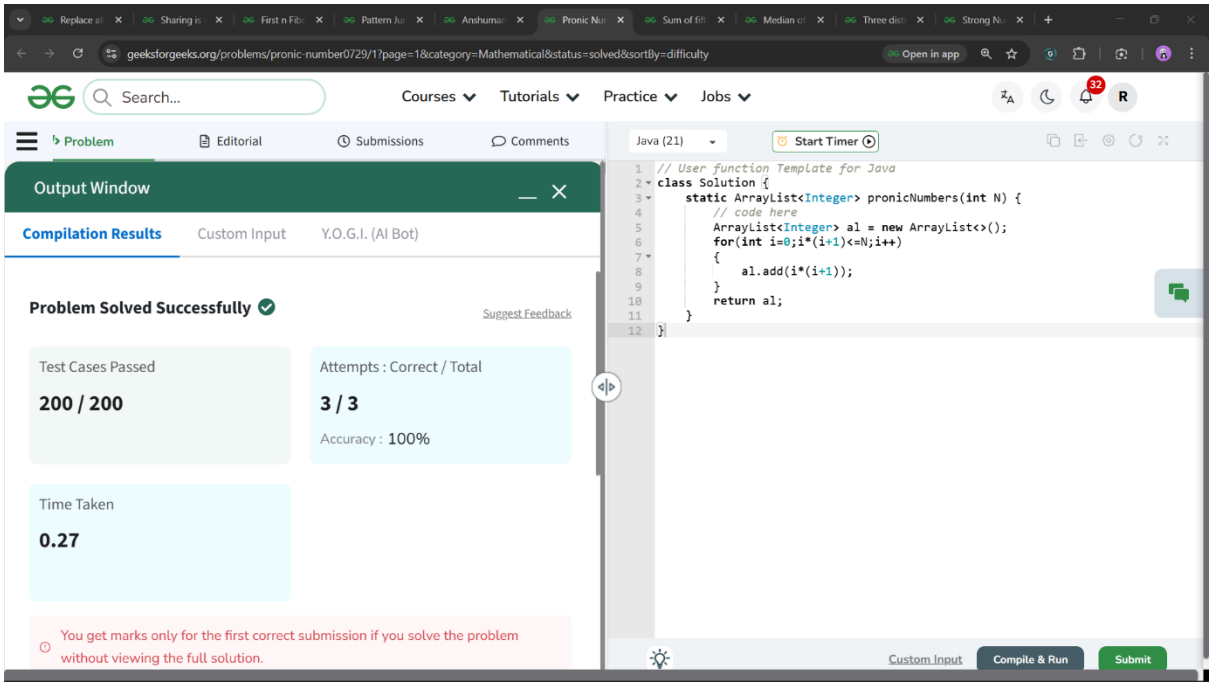
```
class Solution {
   long sumOfFifthPowers(long N) {
      long s=0;
      for(long i=1;i<=N;i++)
      {
         s=s+(long)Math.pow(i,5);
      }
      return s;
   }
}
```



5 . Pronic Number

A pronic number is a number which is the product of two consecutive integers. Find all Pronic Numbers less than or equal to the given integer **N**.
The first few Pronic numbers are: 0, 2, 6, 12, 20, 30, 42, 56, 72, 90, 110, 132 and so on.

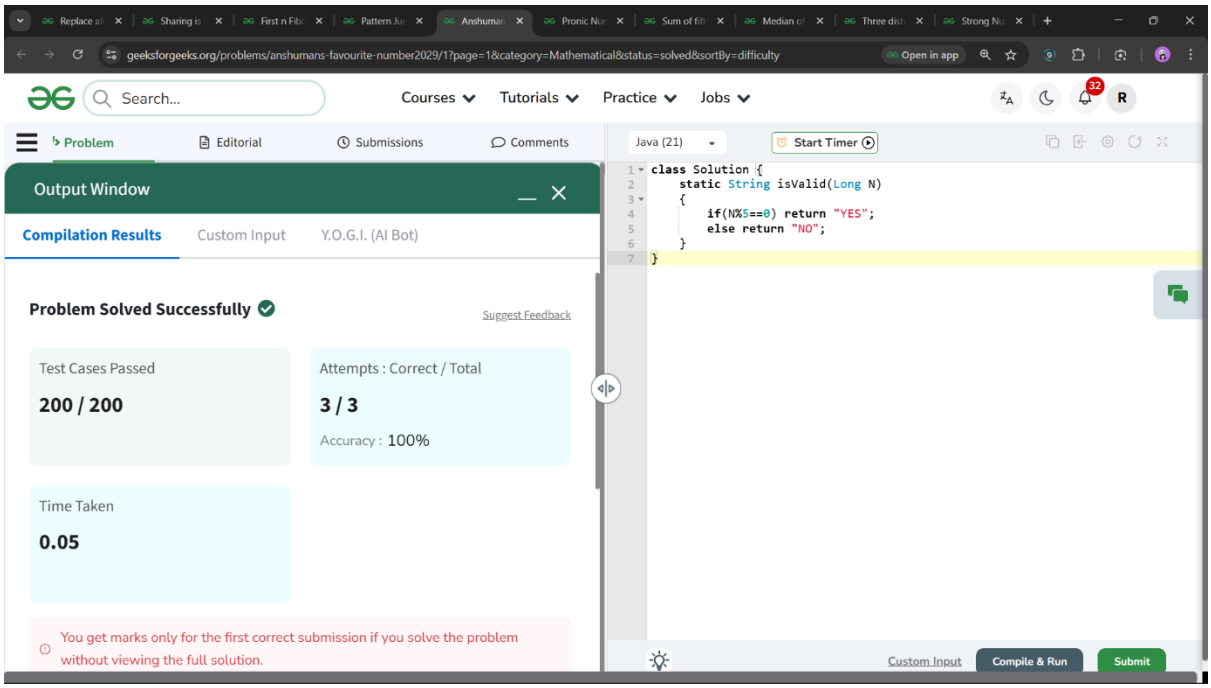
```
class Solution {
   static ArrayList<Integer> pronicNumbers(int N) {
      ArrayList<Integer> al = new ArrayList<>();
      for(int i=0;i*(i+1)<=N;i++)
      {
         al.add(i*(i+1));
      }
      return al;
   }
}
```



6 . Anshuman's Favourite Number

You are given an integer input N and you have to find whether it is the sum or the difference of the integer 5. (5+5, 5+5+5, 5-5,5-5-5+5+5…..)

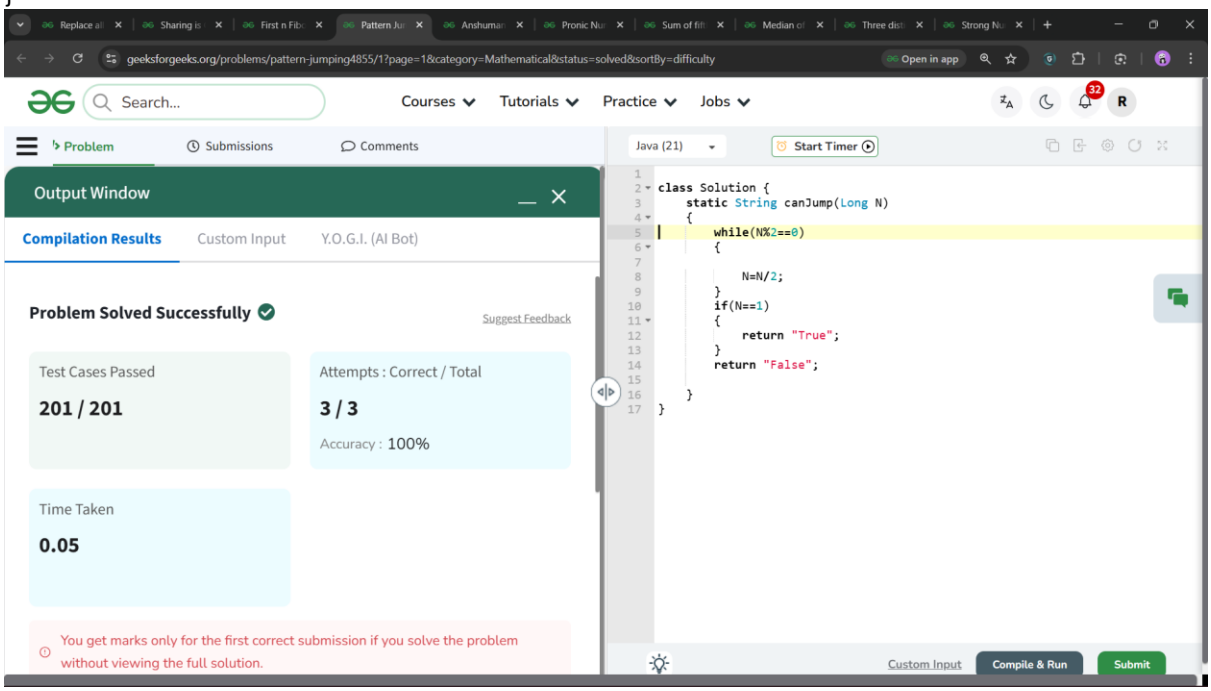
```
class Solution
{
   static String isValid(Long N)
   {
      if(N%5==0) return "YES";
      else return "NO";
   }
}
```



7. Pattern Jumping

A frog starts at the point 0. In his first turn, he can make a jump of 1 unit. Now for all consequent turns, if the frog is currently at a distance x (from the start), his jump will take him x units forward. Given a leaf at a distance **N**, you have to find if the frog can reach that leaf or not.

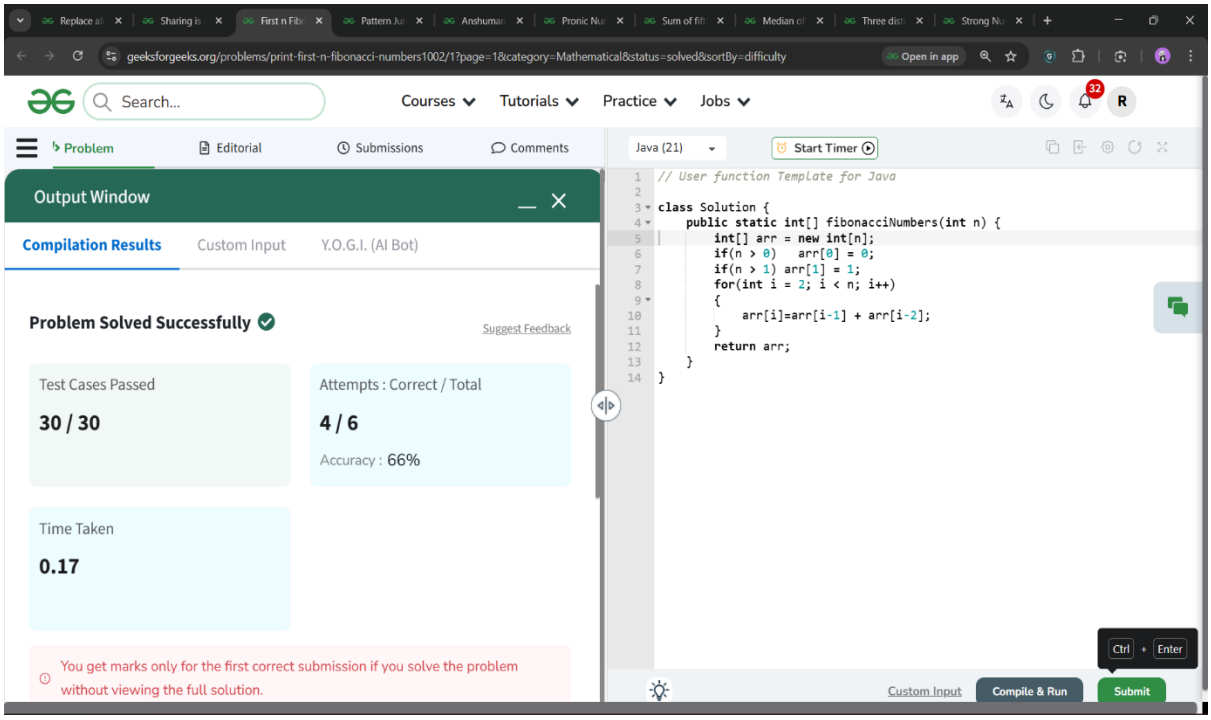
```
class Solution {
   static String canJump(Long N)
   {
      while(N%2==0)
      {

         N=N/2;
      }
      if(N==1)
      {
         return "True";
      }
      return "False";

   }
}
```



8 . First n Fibonacci

Given a number **n,** return an array containing the first **n** Fibonacci numbers.
Note: The first two numbers of the series are 0 and 1.

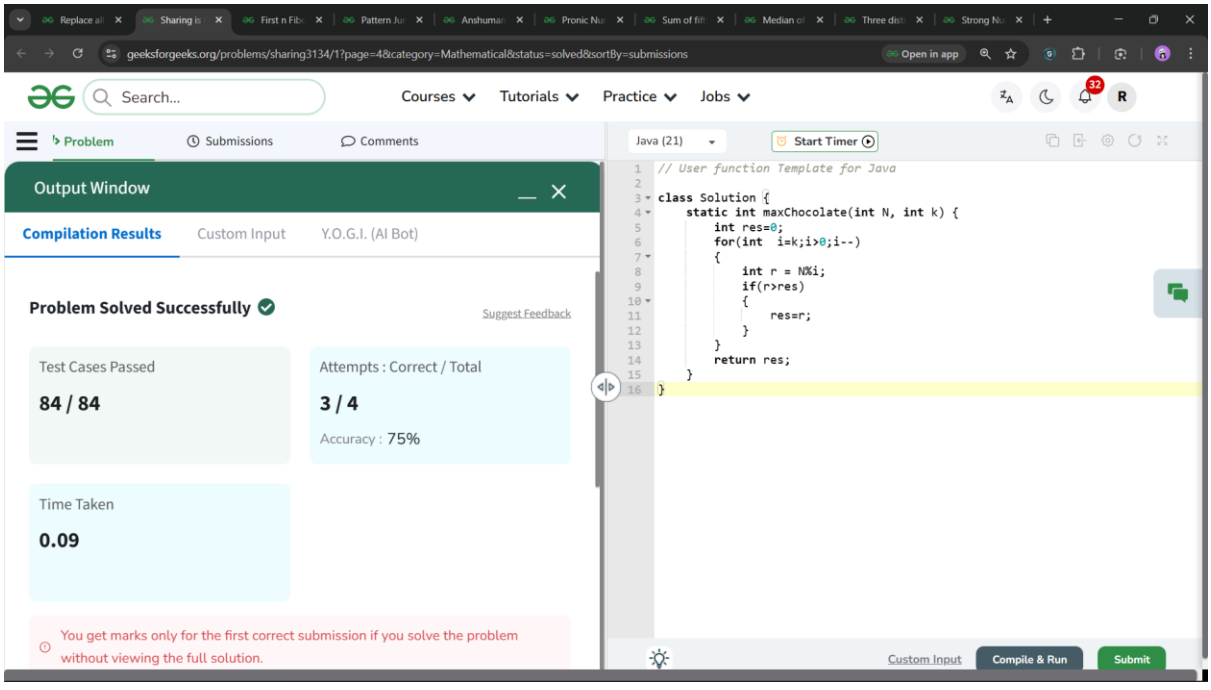
```
class Solution
{
   public static int[] fibonacciNumbers(int n)
   {
      int[] arr = new int[n];
      if(n > 0)   arr[0] = 0;
      if(n > 1) arr[1] = 1;
      for(int i = 2; i < n; i++)
      {
         arr[i]=arr[i-1] + arr[i-2];
      }
      return arr;
   }
}
```



9 . Sharing is Caring

Geek is very fond of chocolates. But he can't reach the kitchen shelf which has **'N'** chocolates. His family has K members and he can call any number of family members to help him out. After acquiring the chocolates, the family members that Geek called will first divide the chocolates amongst themsleves equally. They do this in such a way that they all get maximum number of chocolates. The remaining chocolates are given to Geek.
Find the maximum number of chocolates that Geek can get.

```
class Solution {
   static int maxChocolate(int N, int k) {
      int res=0;
      for(int  i=k;i>0;i--)
      {
         int r = N%i;
         if(r>res)
         {
            res=r;
         }
      }
      return res;
   }
}
```



10 . Replace all 0's with 5

You are given an integer **n**. You need to convert all zeroes of **n** to 5.

```
class Solution {
   int convertfive(int num)
   {
      int n=num;
      int x=5;
      if(n==0)
      return 5;
      while(n>0)
      {
         if(n%10==0)
         {
            num=num+x;
         }
         n=n/10;
         x=x*10;
      }
      return num;

   }
}
```