**SRI VASAVI ENGINEERING COLLEGE**

**PEDATADEPALLI, TADEPALLIGUDEM.**





This is to certify that this is a bonafide record of Practical Work done in **Masters in coding and competitive Programming-I** by Mr./Miss O.Dhanush bearing Roll No .22A81A05N7 of CSE branch during the academic year 2024-2025.

**No. of Experiments Done:60**

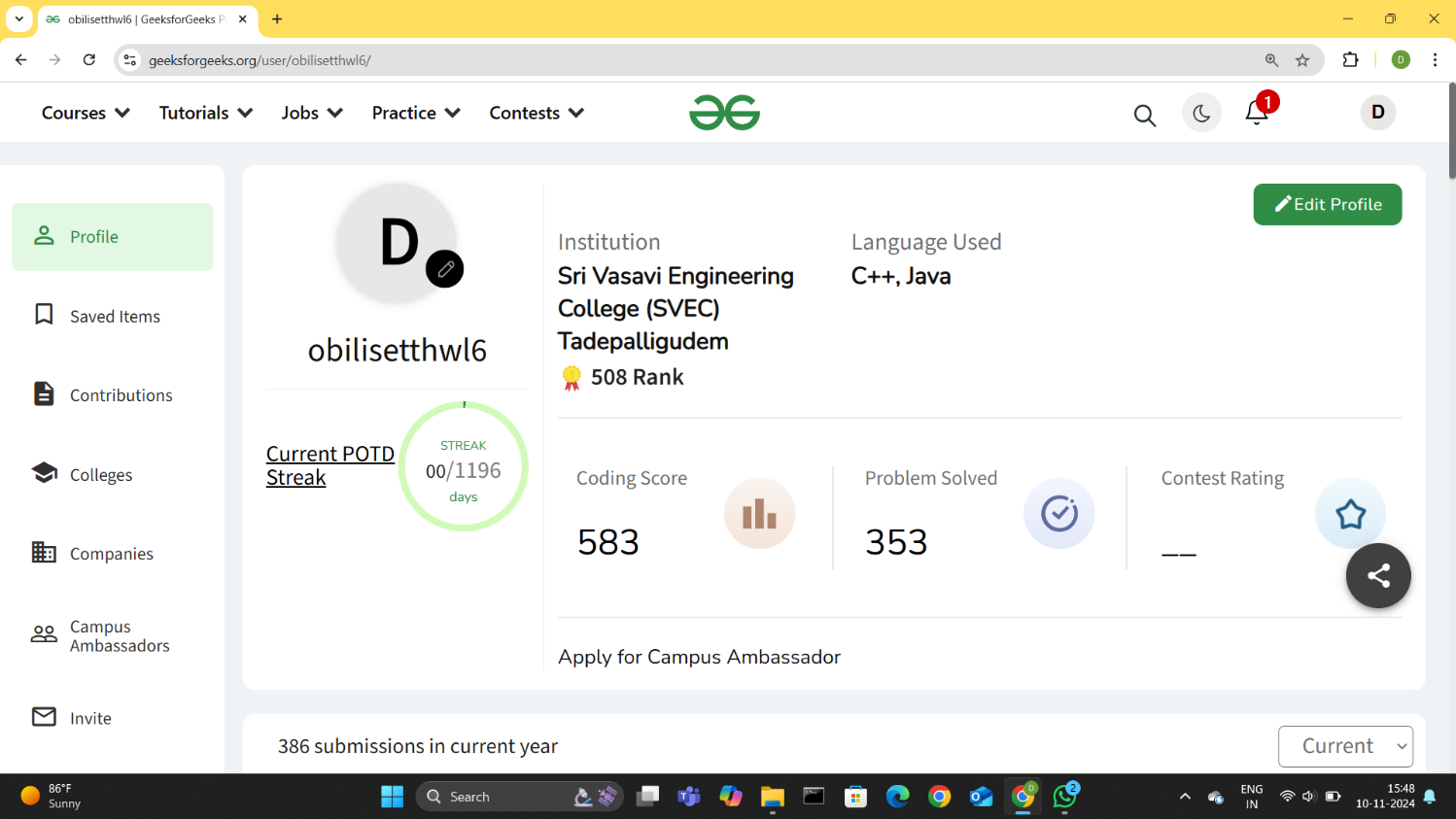
**Faculty In-charge Head of the Department**

**External Examiner**

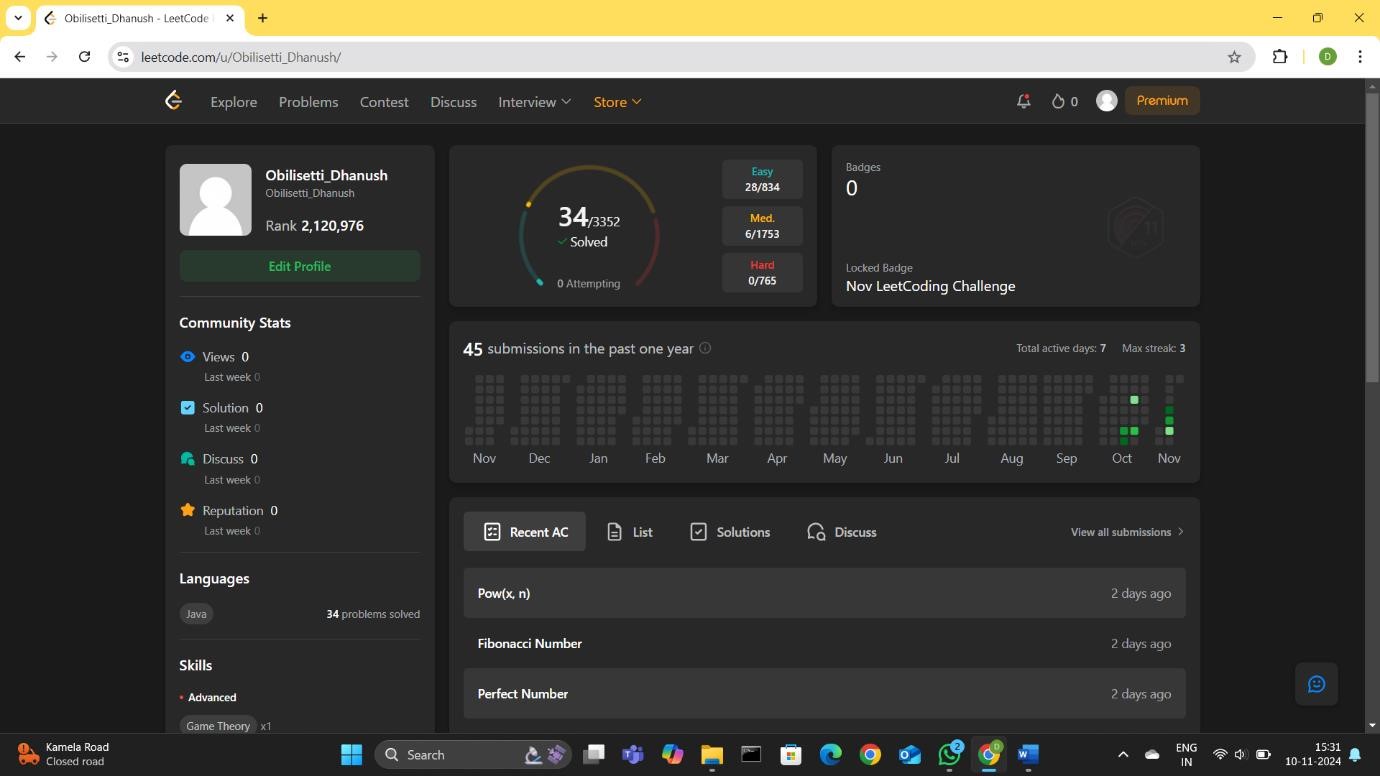
# Index

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment No** | **Experiment Name** | **Date** | **Signature** |
| 1 | Square Root of a number |  |  |
| 2 | LCM and GCD |  |  |
| 3 | Sum of first n terms |  |  |
| 4 | Count squares |  |  |
| 5 | Odd or even |  |  |
| 6 | Missing numbers |  |  |
| 7 | GCD of two numbers |  |  |
| 8 | Reverse digits |  |  |
| 9 | Swap two numbers |  |  |
| 10 | Multiplication table |  |  |
| 11 | Prime number |  |  |
| 12 | Largest Prime factor |  |  |
| 13 | Prime pair with largest sum |  |  |
| 14 | Find prime numbers in a range |  |  |
| 15 | Transform to prime |  |  |
| 16 | Prime factors |  |  |
| 17 | Primes sum |  |  |
| 18 | Prime string |  |  |
| 19 | Damon prime |  |  |
| 20 | Semi prime |  |  |
| 21 | Nth Fibonacci number |  |  |
| 22 | Fibonacci series upto nth term |  |  |
| 23 | Fibonacci sum |  |  |
| 24 | Nth even Fibonacci number |  |  |
| 25 | Check if the number is Fibonacci |  |  |
| 26 | Series AP |  |  |
| 27 | Difference Series |  |  |
| 28 | Crack the series |  |  |
| 29 | Series X1 |  |  |
| 30 | Mansi and her series |  |  |
| 31 | Missing in array |  |  |
| 32 | Sort o’s,1’s. and 2’s |  |  |
| 33 | Binary search |  |  |
| 34 | Array search |  |  |
| 35 | Array subset |  |  |
| 36 | Largest element in array |  |  |
| 37 | Rotate array by one |  |  |
| 38 | Remove duplicates Sorted array |  |  |
| 39 | Wave array |  |  |
| 40 | Product array puzzle |  |  |
| 41 | Balanced Array |  |  |
| 42 | Alternates in an array |  |  |
| 43 | Palindromic array |  |  |
| 44 | Adding array elements |  |  |
| 45 | Last index of one |  |  |
| 46 | Reverse a string |  |  |
| 47 | Palindrome string |  |  |
| 48 | Reverse words |  |  |
| 49 | Isomorphic strings |  |  |
| 50 | Implement strstr |  |  |
| 51 | Maximum occurring character |  |  |
| 52 | Remove consecutive characters |  |  |
| 53 | Repeated character |  |  |
| 54 | Convert string to lower case |  |  |
| 55 | Display longest name |  |  |
| 56 | Remove common characters and concatenate |  |  |
| 57 | Sum of numbers in string |  |  |
| 58 | Java strings | set 1 |  |  |
| 59 | C++ strings |  |  |
| 60 | Pattern searching |  |  |

**GeeksForGeeks Profile:**

****

**LeetCode Profile:**



**Mathematics:**

## 1.Square root of a number

class Solution { public:

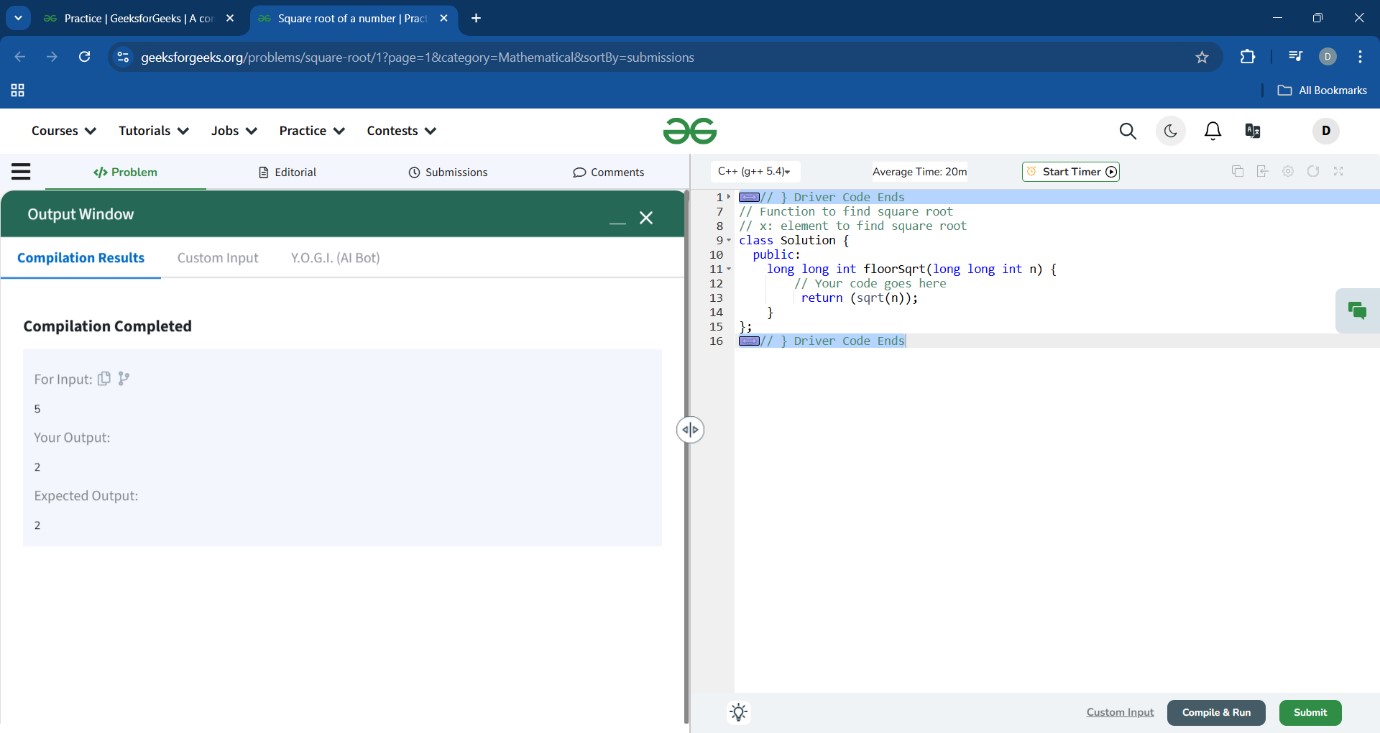
long long int floorSqrt(long long int n) {

// Your code goes here return (sqrt(n));

}

}

;



## 2.LCM And GCD

class Solution { public:

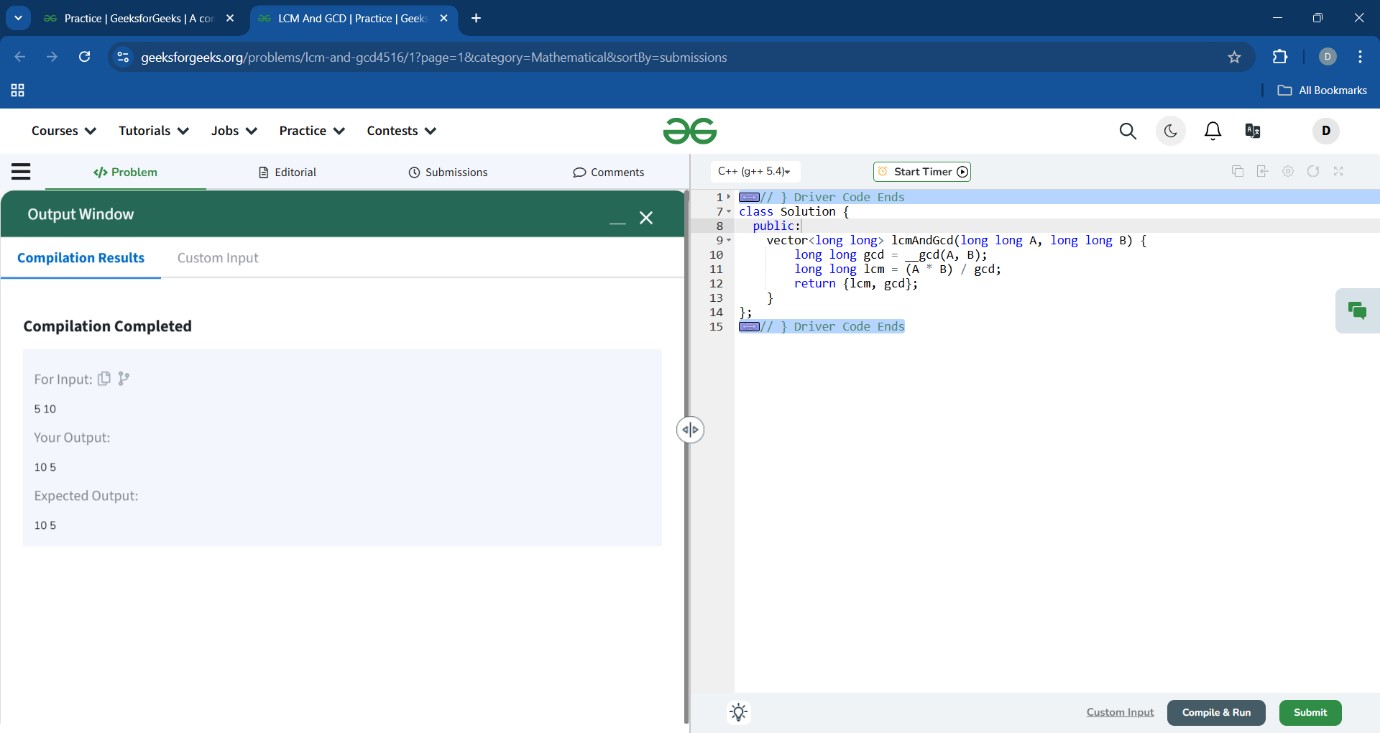
vector<long long> lcmAndGcd(long long A, long long B) { long long gcd = \_\_gcd(A, B); long long lcm = (A \* B) / gcd; return {lcm, gcd};

}

};

**3**

**.Sum of first n terms**



class Solution { public:

long long sumOfSeries(long long n) {

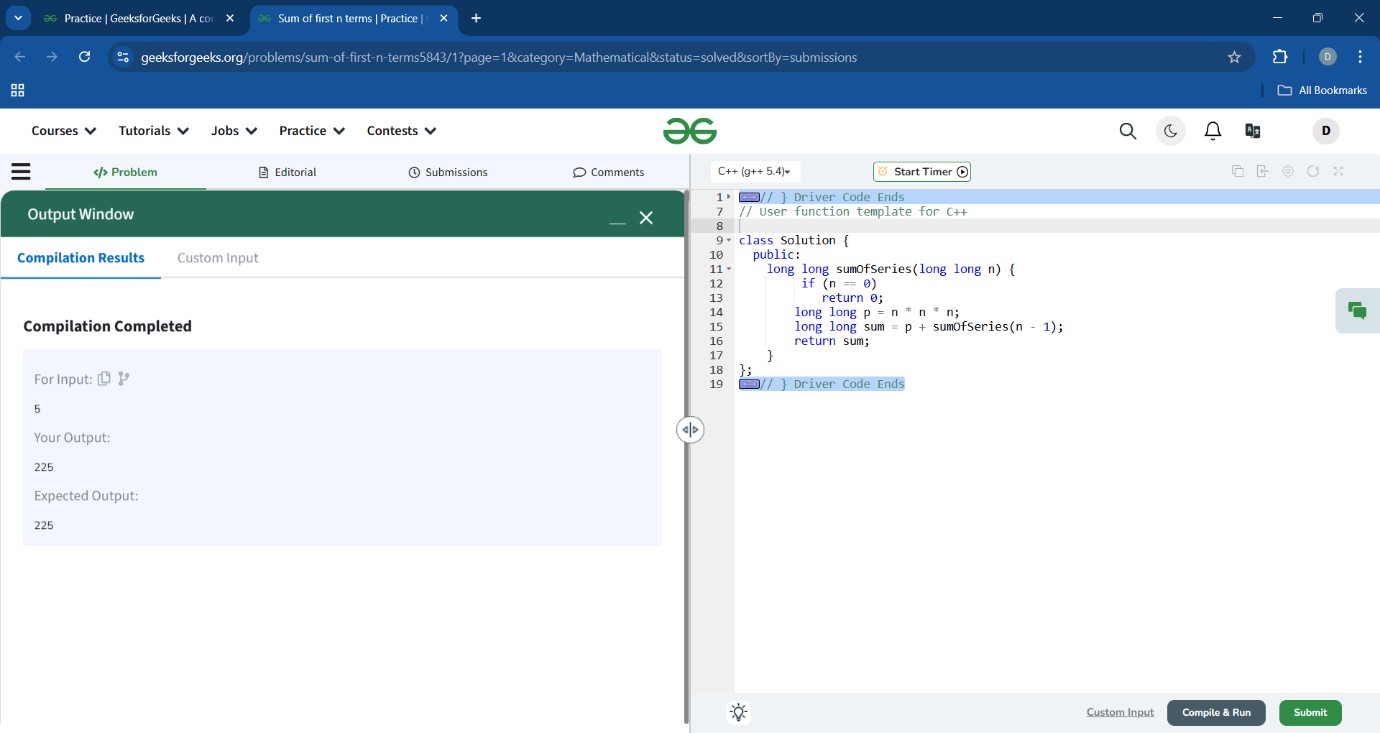
if (n == 0) return 0; long long p = n \* n \* n; long long sum = p + sumOfSeries(n - 1); return sum;

}

};

**4.**

**Count Squares**



class Solution { public: int countSquares(int N) { int count=0; int i=2; int square=1; while(square<N){ square=i\*i; count++; i++;

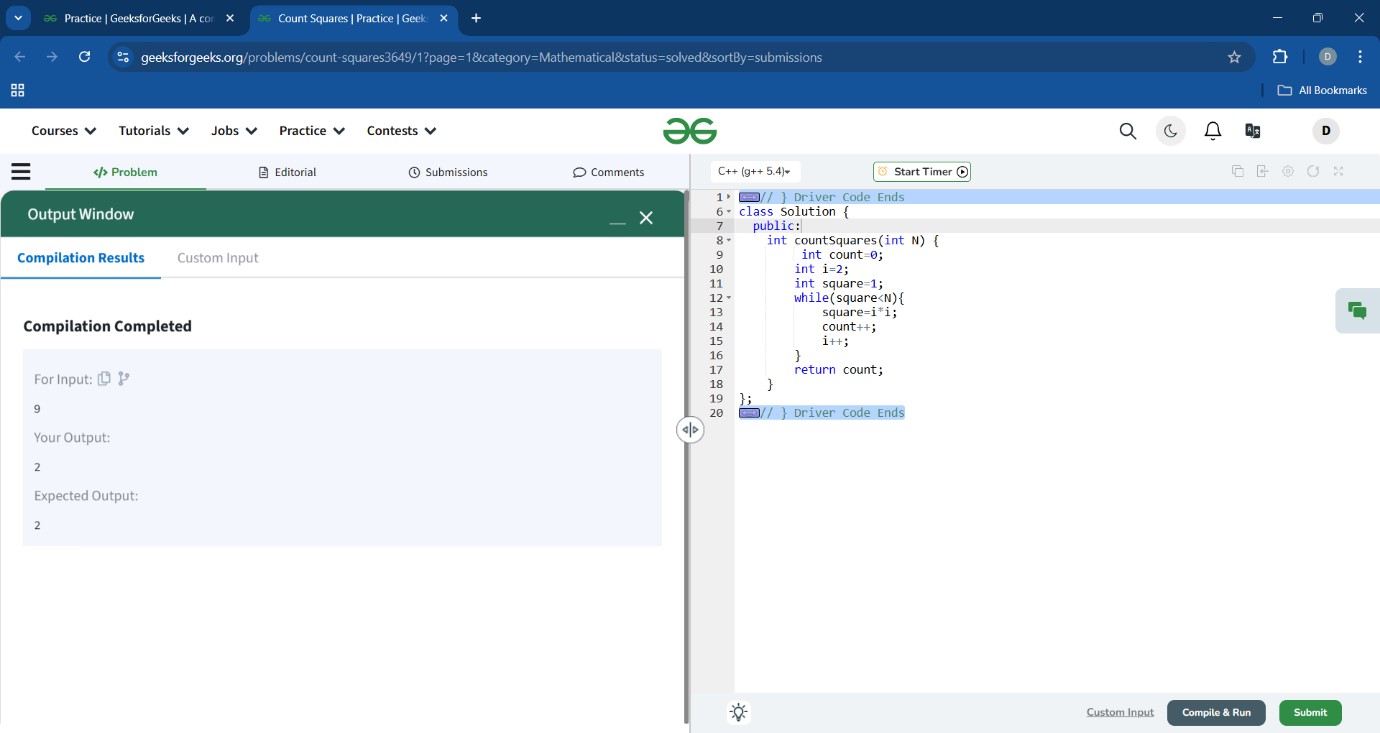
} return count;

}

};

**5**

**.Odd or Even**



class Solution { public: string oddEven(int n) { if(n%2==0) { return "even";

} else { return "odd";

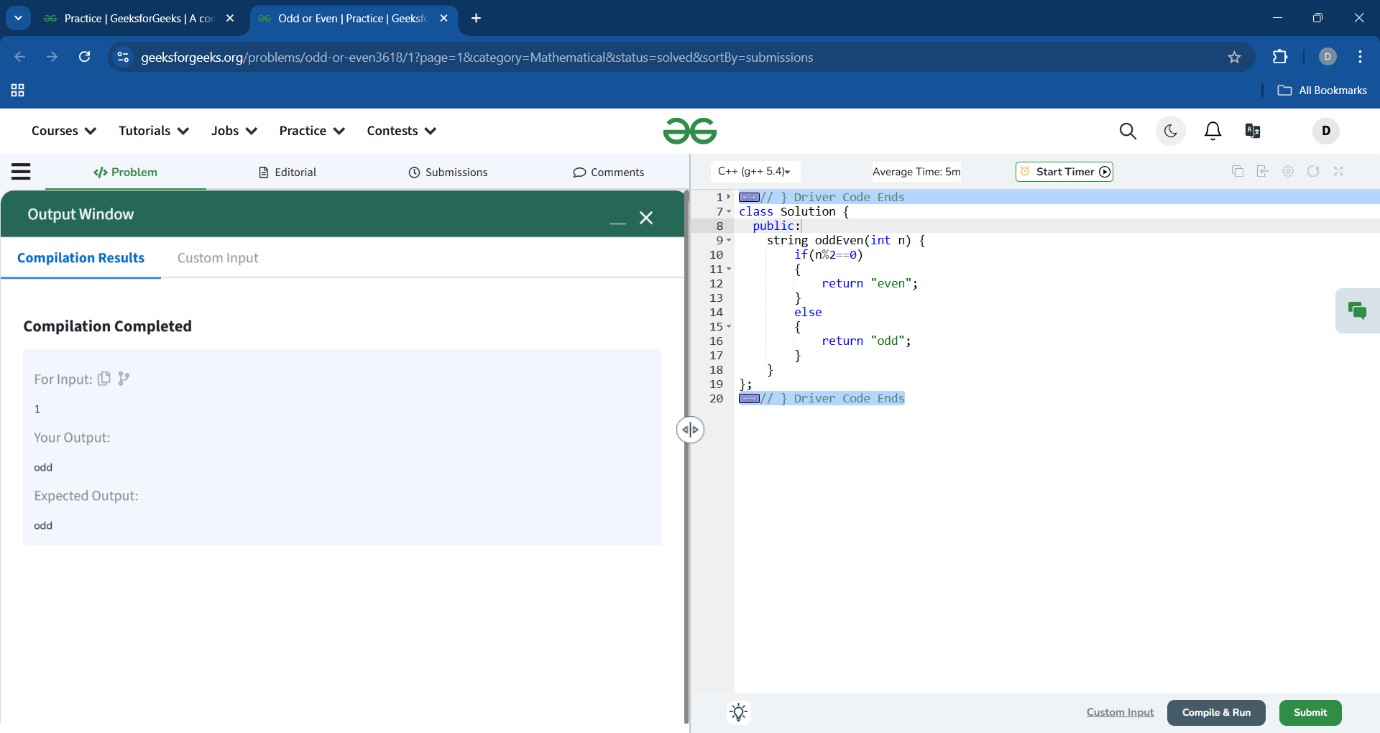
}

}

};

**6**

**.Missing Number**



class Solution { public static int missingNumber(int n, int[] arr) { int s=0; for(int i=1;i<=n;i++)

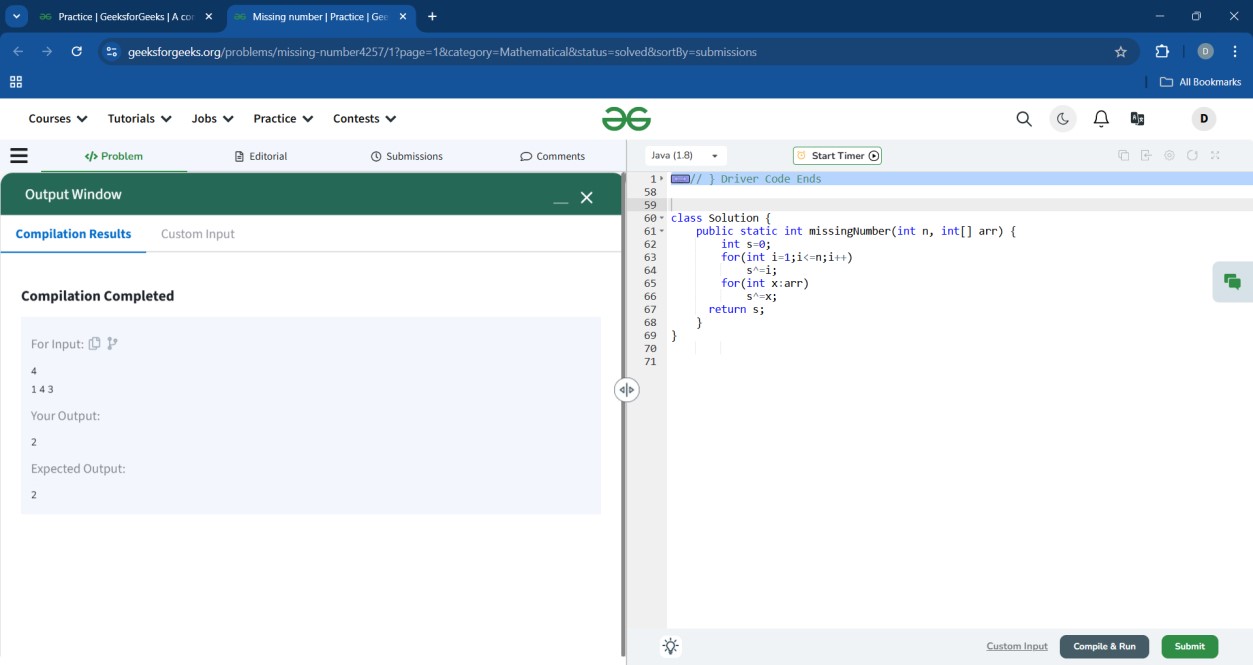
s^=i; for(int x:arr) s^=x; return s;

}

}

**7**

**.GCD of two numbers**



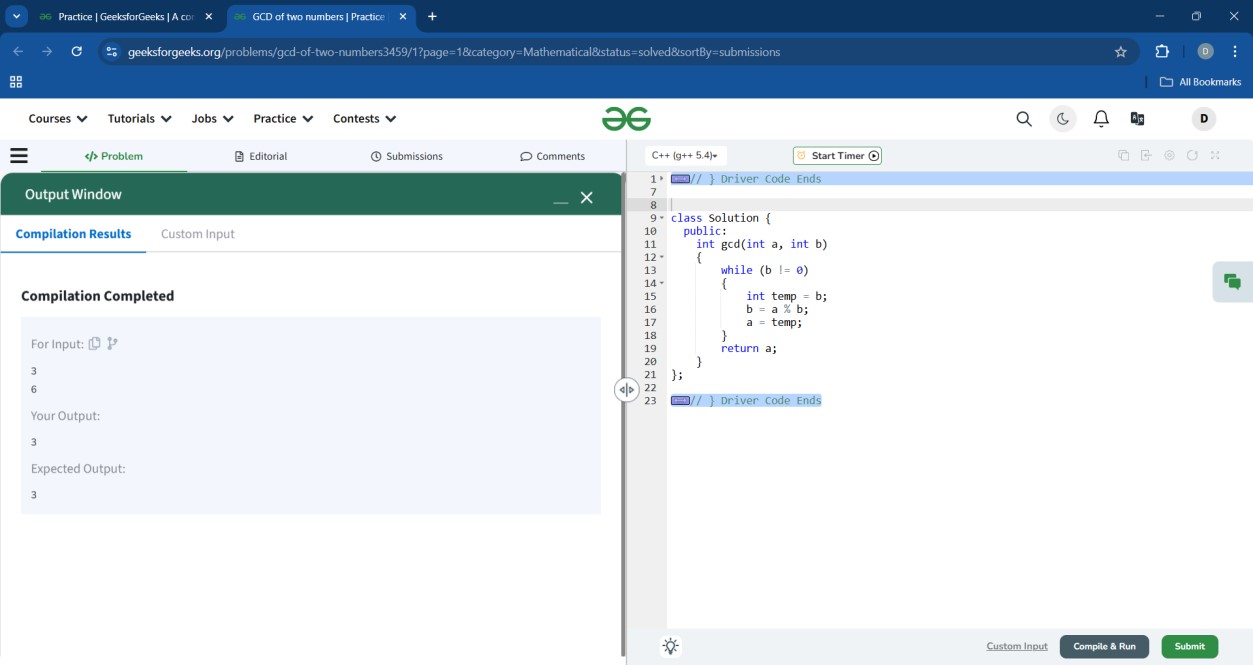
class Solution { public: int gcd(int a, int b) { while (b != 0)

{ int temp = b; b = a % b; a = temp; } return a; }

};

**8**

**.Reverse digits**



class Solution

{

public:

long long int reverse\_digit(long long int n)

{

string s=to\_string(n); string s1="";

for(int i=s.size()-1;i>=0;i--)

{

s1=s1+s[i];

}

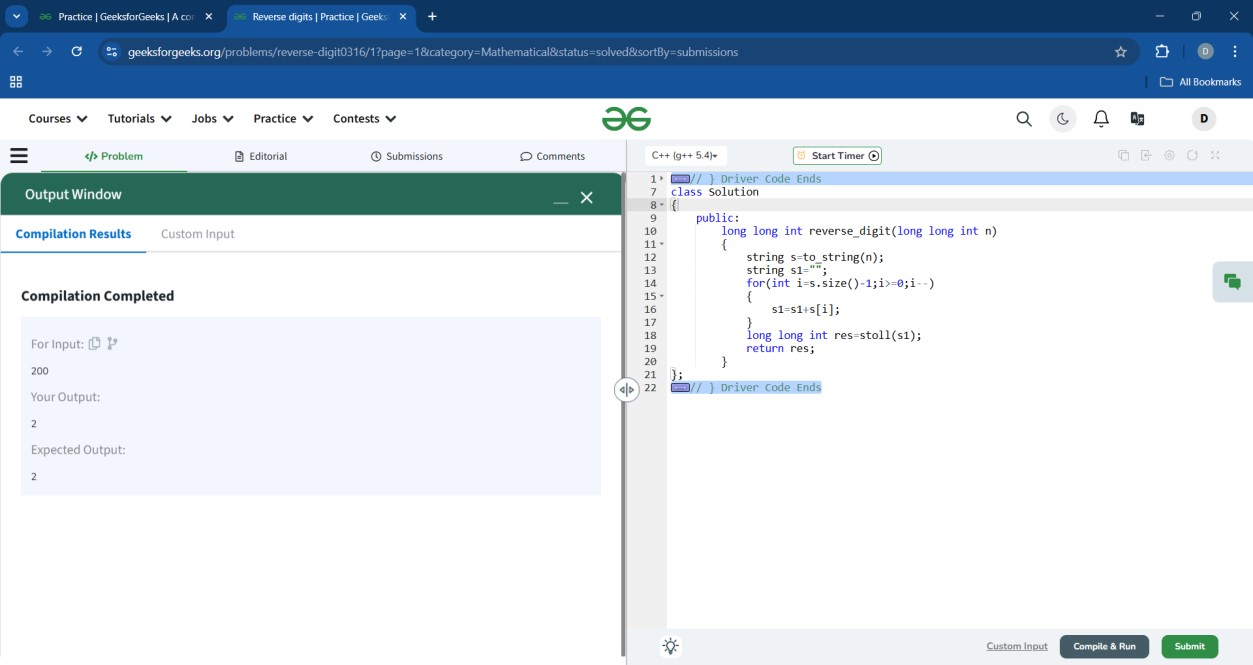
long long int res=stoll(s1); return res;

}

};

**9**

**. Swap two numbers**



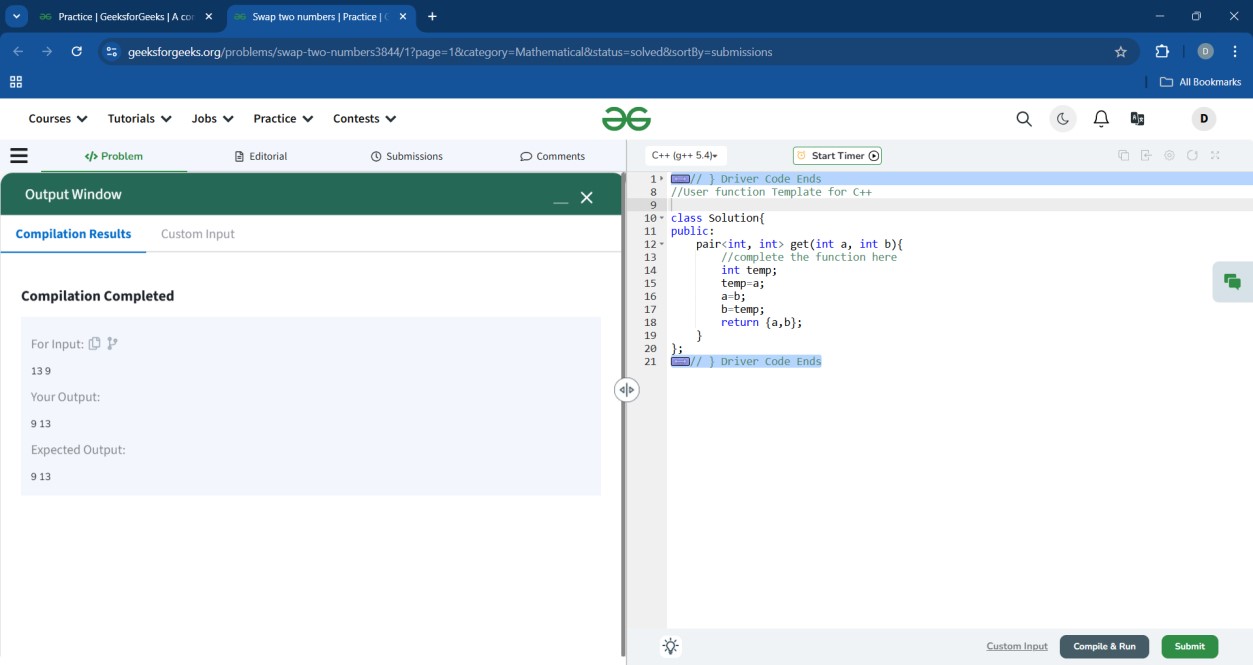
class Solution{ public: pair<int, int> get(int a, int b){ int temp; temp=a; a=b; b=temp; return {a,b};

}

};

**10**

**.Multiplication table**



class Solution { public: vector<int> getTable(int N)

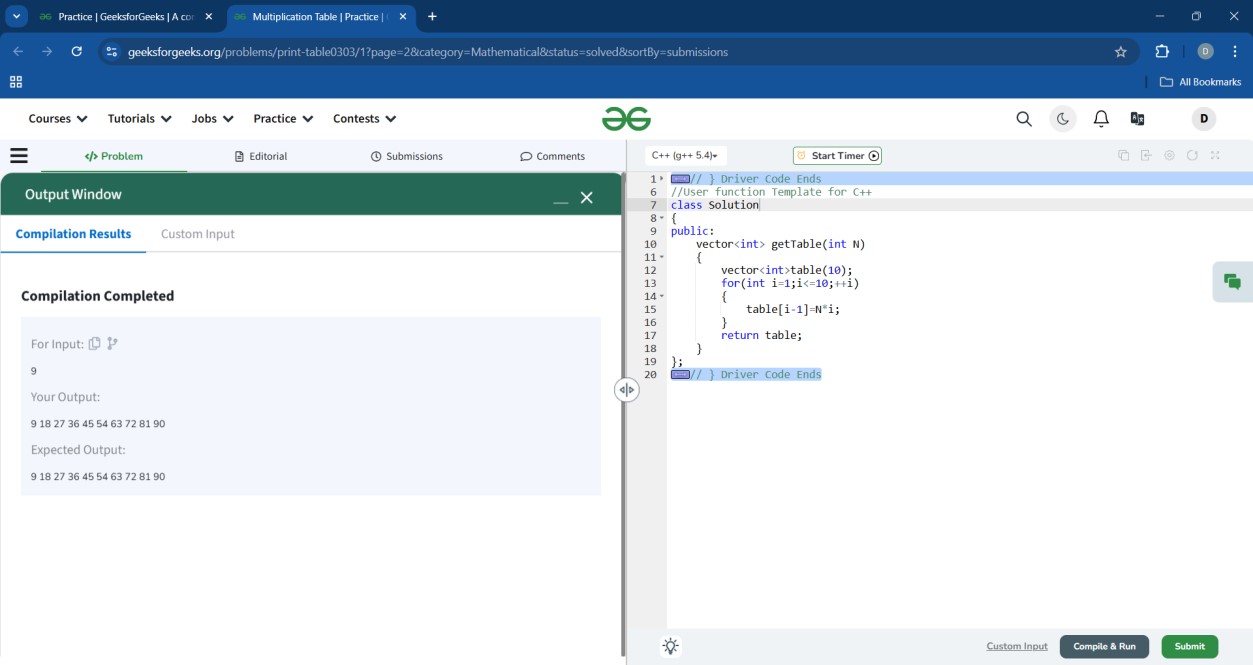
{ vector<int>table(10); for(int i=1;i<=10;++i) { table[i-1]=N\*i;

} return table;

}

};

**Prime numbers**



## 1.Prime number

class Solution{ public: int isPrime(int N){ if (N <= 1) { return 0;

} for (int i = 2; i <= sqrt(N); i++) { if (N % i == 0) { return 0;

} } return 1;

}

};

**2**

**.Largest Prime factor**



class Solution{ public: long long int largestPrimeFactor(int n) { long long int largest = -1; while (n % 2 == 0) { largest = 2; n /= 2;

} for (int i = 3; i \* i <= n; i += 2) { while (n % i == 0) { largest = i; n /= i;

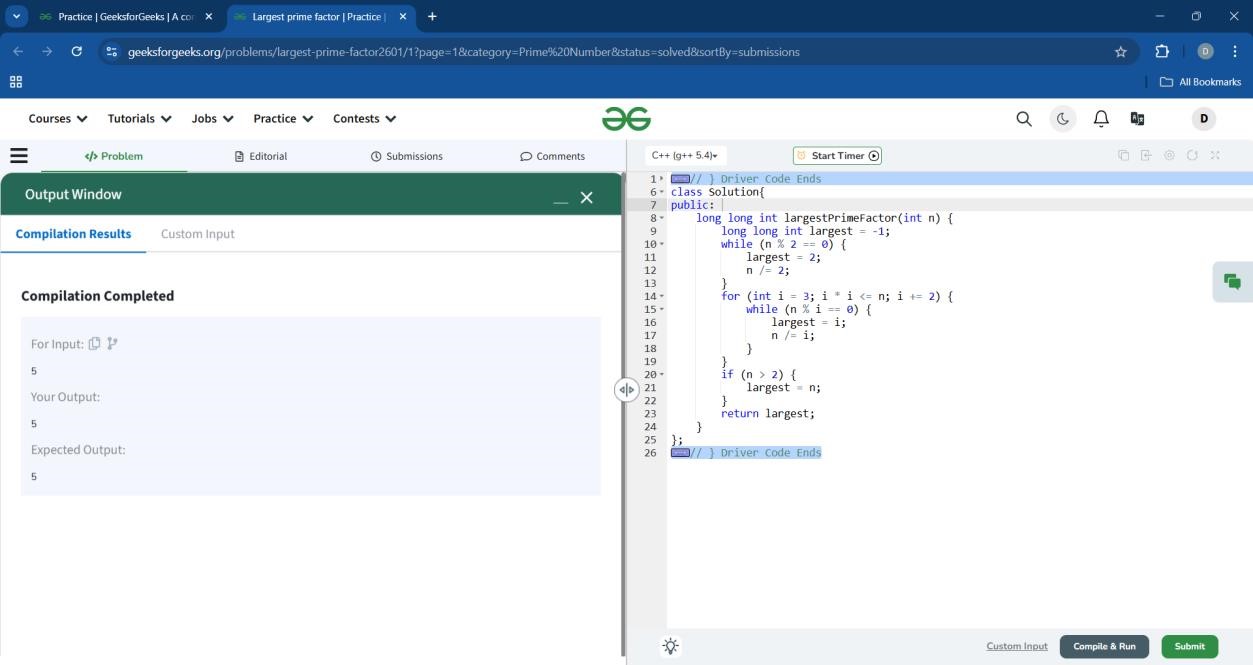
} } if (n > 2) largest = n; return largest;

}

};

**3**

**.Prime pair with target sum**



class Solution { public:

vector<int> getPrimes(int n) { vector<int> v; for(int i =2 ;i<=n/2; i++)

{

if(prime(i)&&prime(n-i))

{

v.push\_back(i);

v.push\_back(n-i); return v;

}

}

v.push\_back(-1);

v.push\_back(-1); return v;

}

bool prime(int n)

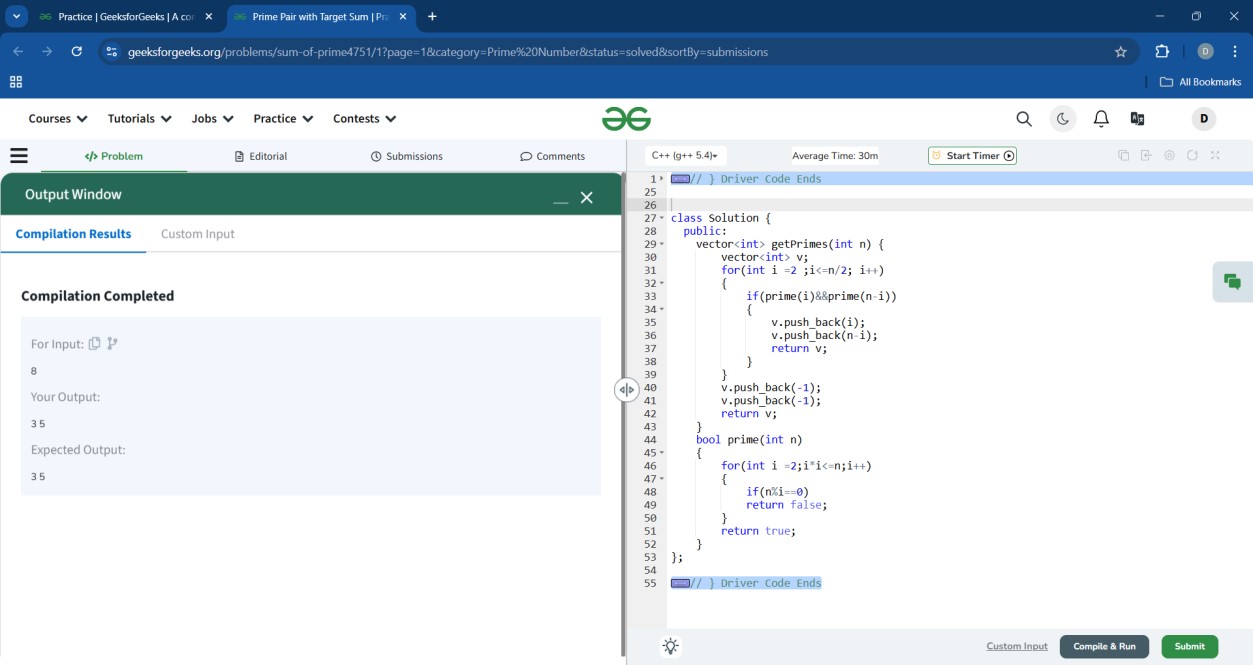
{ for(int i =2;i\*i<=n;i++)

{ if(n%i==0) return false; } return true;

}

}

;



## 4. Find Prime numbers in a range

class Solution { public:

vector<int> primeRange(int M, int N) { vector<int> v; bool a[N+1]; memset(a,true,sizeof(a)); a[0]=0; a[1]=0; for(int i=2;i\*i<=N;i++)

{ if(a[i]==true)

{ for(int j=i\*i;j<=N;j=j+i)

{ a[j]=0;

}

}

}

for(int i=M;i<=N;i++)

{ if(a[i]==true)

{

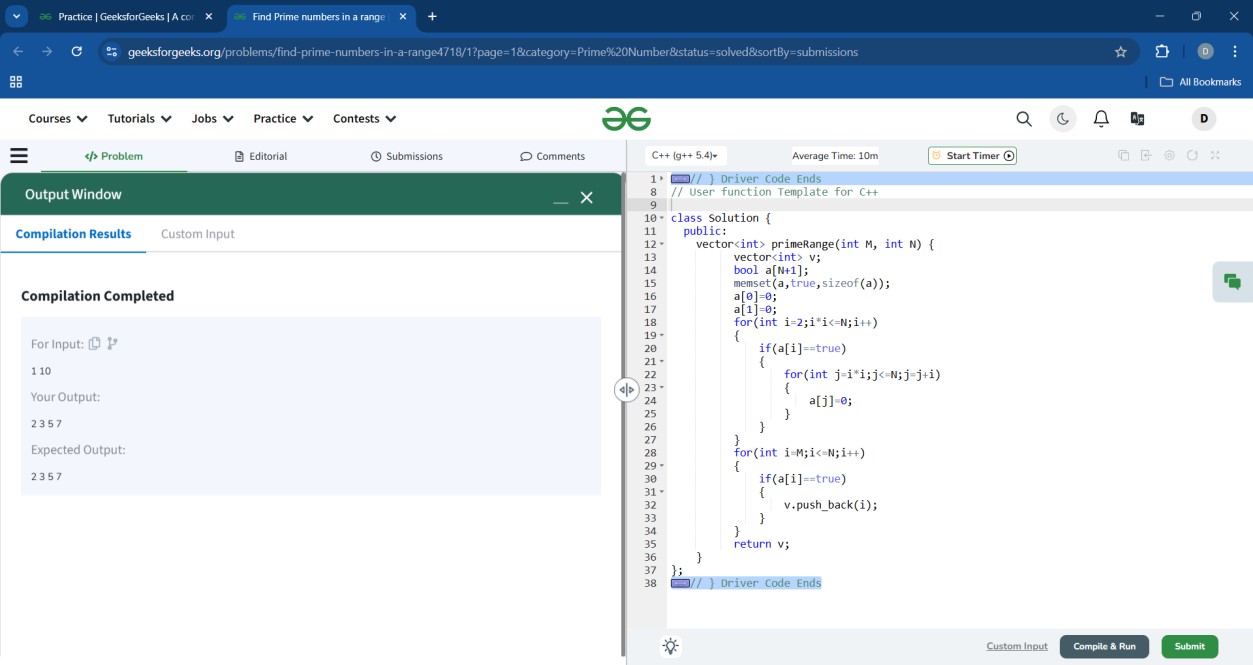
v.push\_back(i);

} } return v;

}

}

;



## 5. Transform to prime

class Solution { public:

bool prime(int sum){ for(int i=2;i\*i<=sum;i++){ if(sum%i==0){ return false;

} } return true;

}

int minNumber(int arr[],int N)

{ int sum=0; for(int i=0;i<N;i++){ sum=sum+arr[i];

}

for(int i=0;i<=sum;i++){ if(sum==1){ return 1;

}

if(prime(sum+i)){

return i;

}

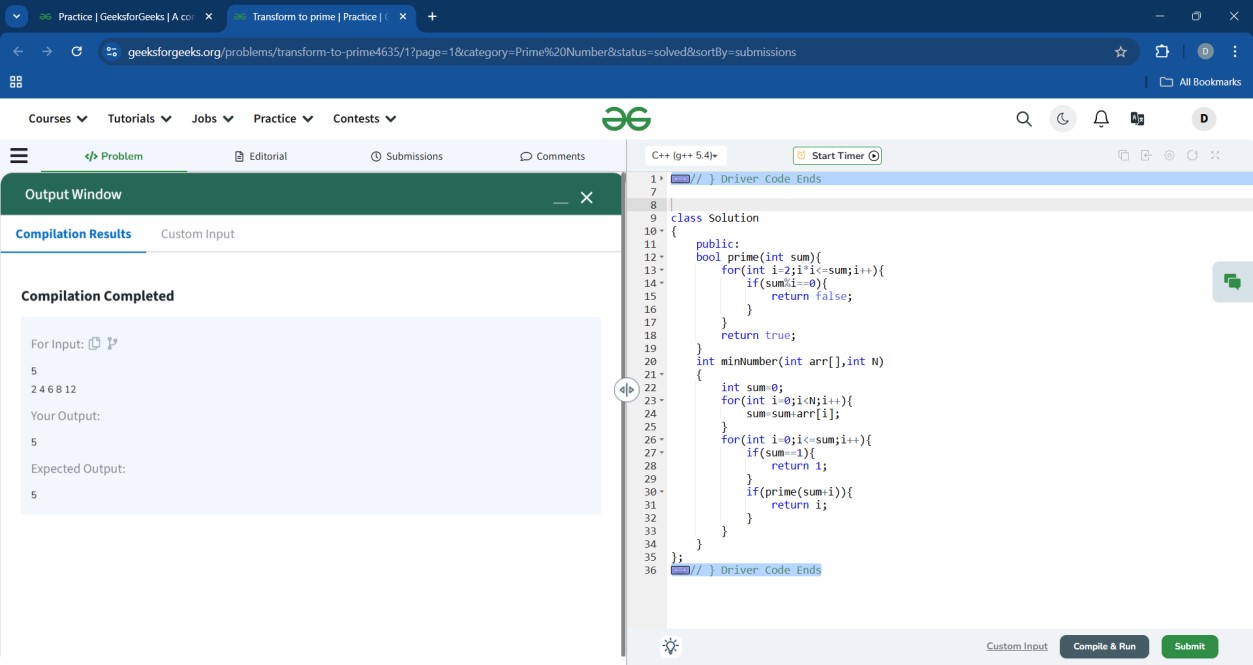
}

}

};

**6.**

**Prime Factors**



class Solution{ public:

vector<int>AllPrimeFactors(int N) { vector<int> ans;

for(int i=2; i<=sqrt(N); i++) { if(N%i==0) { ans.push\_back(i); while(N%i==0) {

N=N/i;

}

}

} if(N!=1)

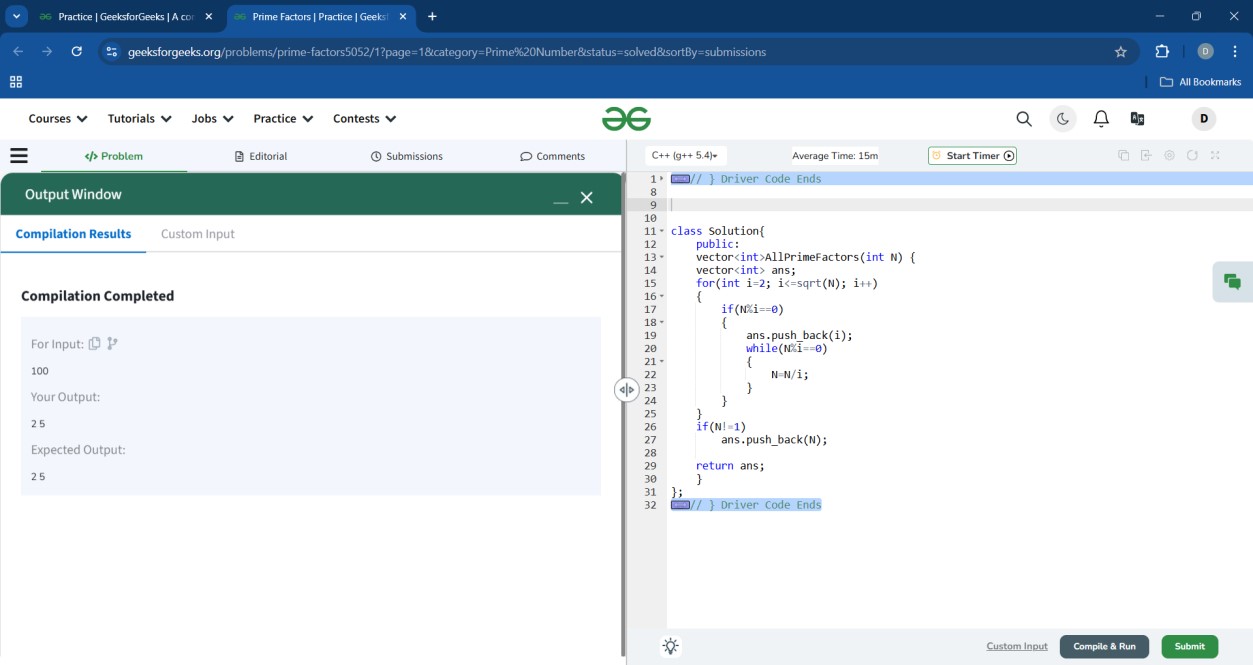
ans.push\_back(N); return ans;

}

};

**7.**

**Primes sum**



class Solution { public: string isSumOfTwo(int N){

if(N <= 2) { return "No";

}

for( int i=2;i<=N/2;++i) { if(isPrime(i)&&isPrime(N-i)) { return "Yes";

}

} return "No";

}

bool isPrime(int n)

{ if(n <= 1) { return false; } for(int i=2 ; i <= sqrt(n) ; i++)

{ if(n % i == 0)

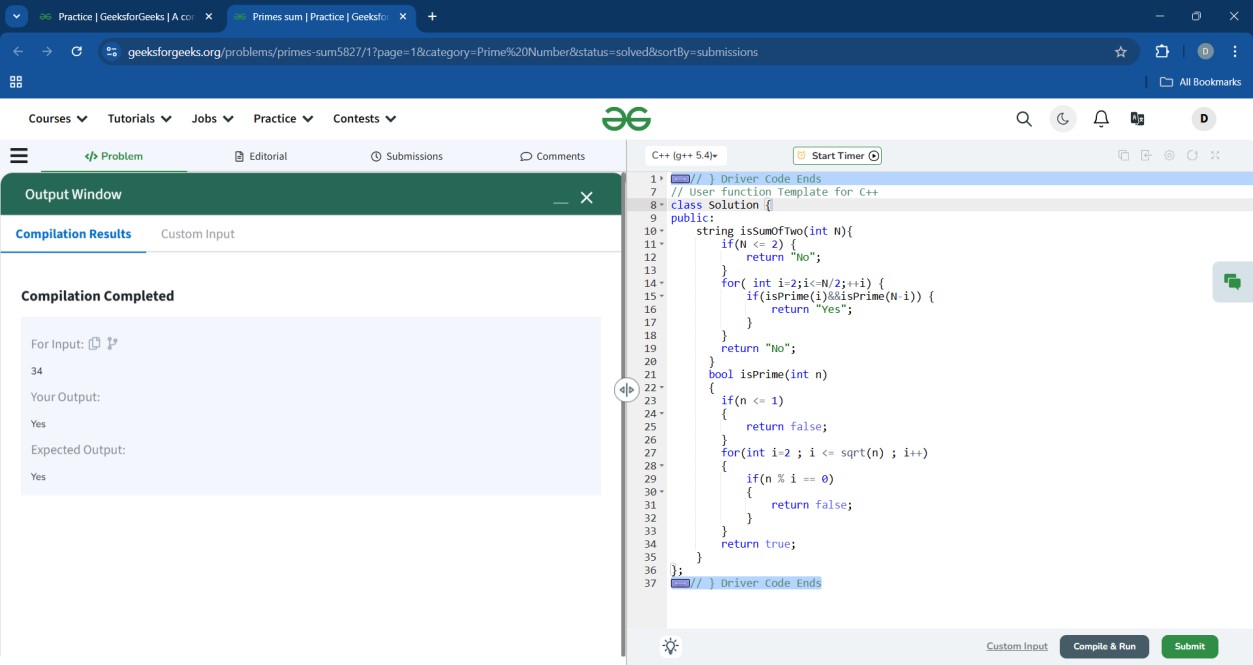
{ return false;

} } return true;

}

}

;



## 8. Prime String

class Solution { public boolean isPrimeString(String s)

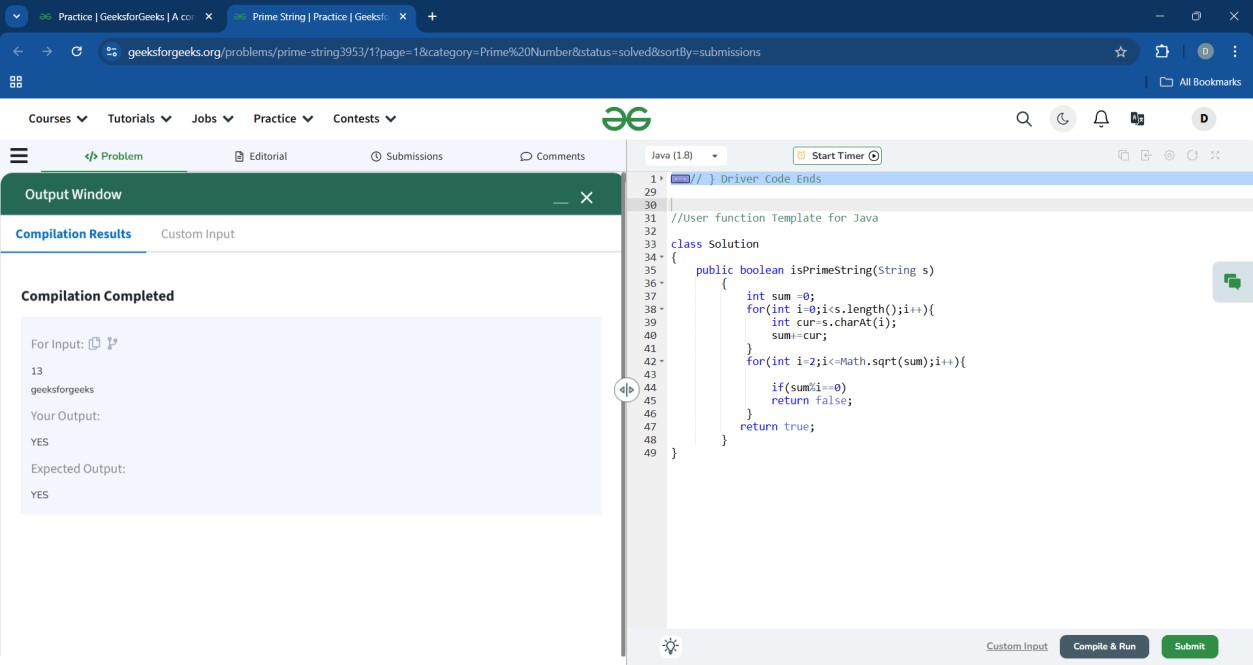
{ int sum =0; for(int i=0;i<s.length();i++){ int cur=s.charAt(i); sum+=cur;

}

for(int i=2;i<=Math.sqrt(sum);i++){ if(sum%i==0) return false; } return true;

}

}



## 9. Damon Prime

class Solution { bool isprime(int n){ if(n<=1) return 0; for(int i=2;i<n;i++)

{ if(n%i==0)

{ return 0; } } return 1;

} public:

string damonPrime(int N){ if(isprime(N-1)&&isprime(N+1))

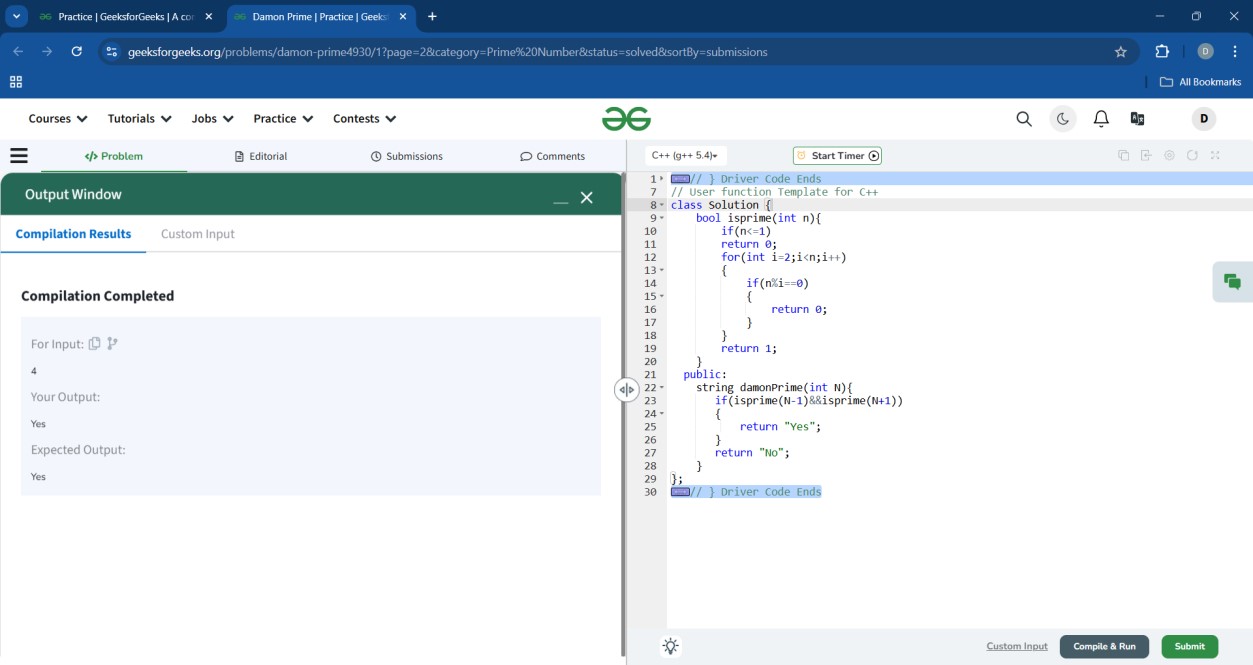
{ return "Yes";

} return "No";

}

}

;



## 10.Semi Prime

class Solution { public: int checkSemiprime(int n) {

int c=0; while(n%2==0) { n=n/2; c++;

}

for(int i=3;i\*i<=n;i=i+2) { while(n%i==0) {

c++; n=n/i;

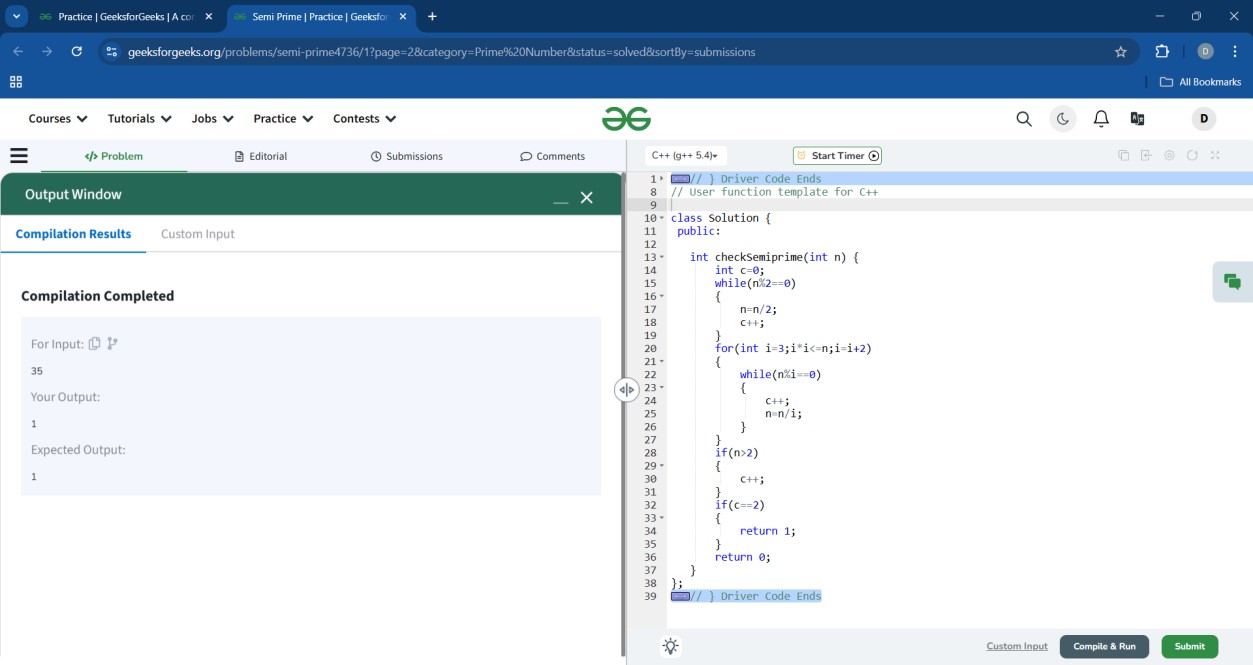
} } if(n>2) { c++;

} if(c==2) { return 1; } return 0;

}

}

;



# Fibonacci Numbers

## 1.Nth Fibonacci number

class Solution { public: int nthFibonacci(int n){ // code here int a=1,b=1,c; if(n==0)

{ return 0; } if(n==1)

{ return a;

} else if(n==2)

{ return b;

}

for(int i=3;i<=n;i++)

{

c=(a+b)%1000000007;

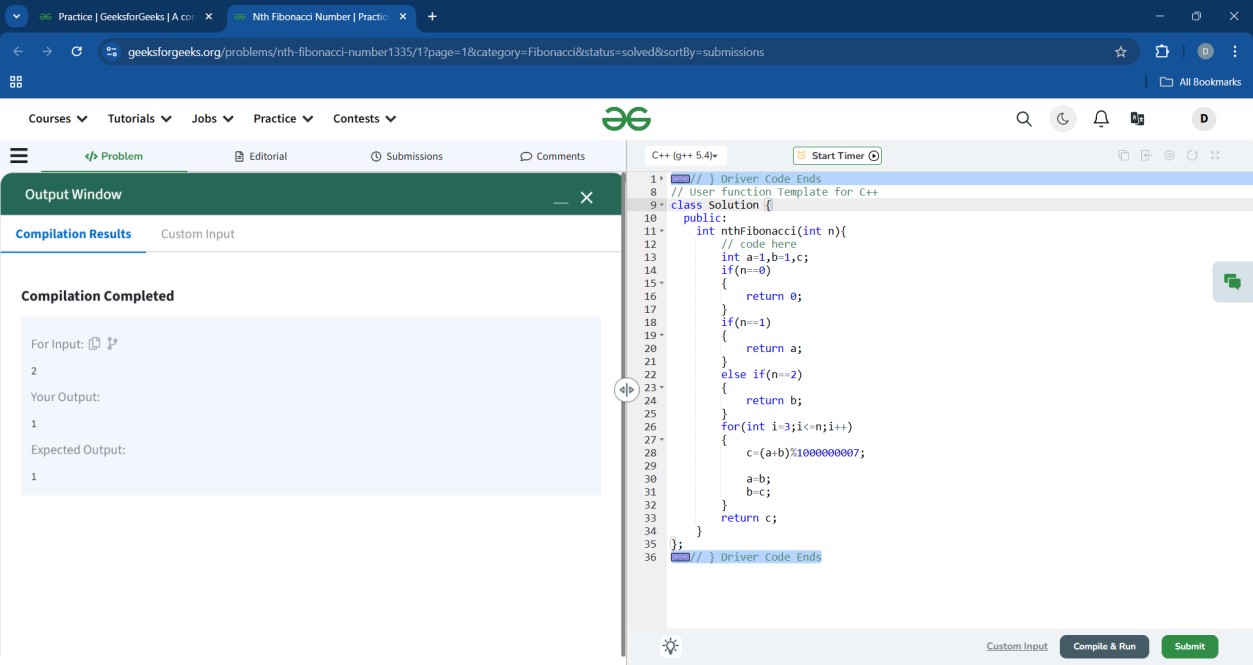
a=b; b=c; } return c;

}

};

**2.**

**Fibonacci series up to Nth term**



class Solution { public: vector<int> Series(int n) { vector<int>v; int a=0,b=1,c;

v.push\_back(0);

v.push\_back(1); for(int i=1;i<n;i++)

{

c=(a+b)%1000000007;

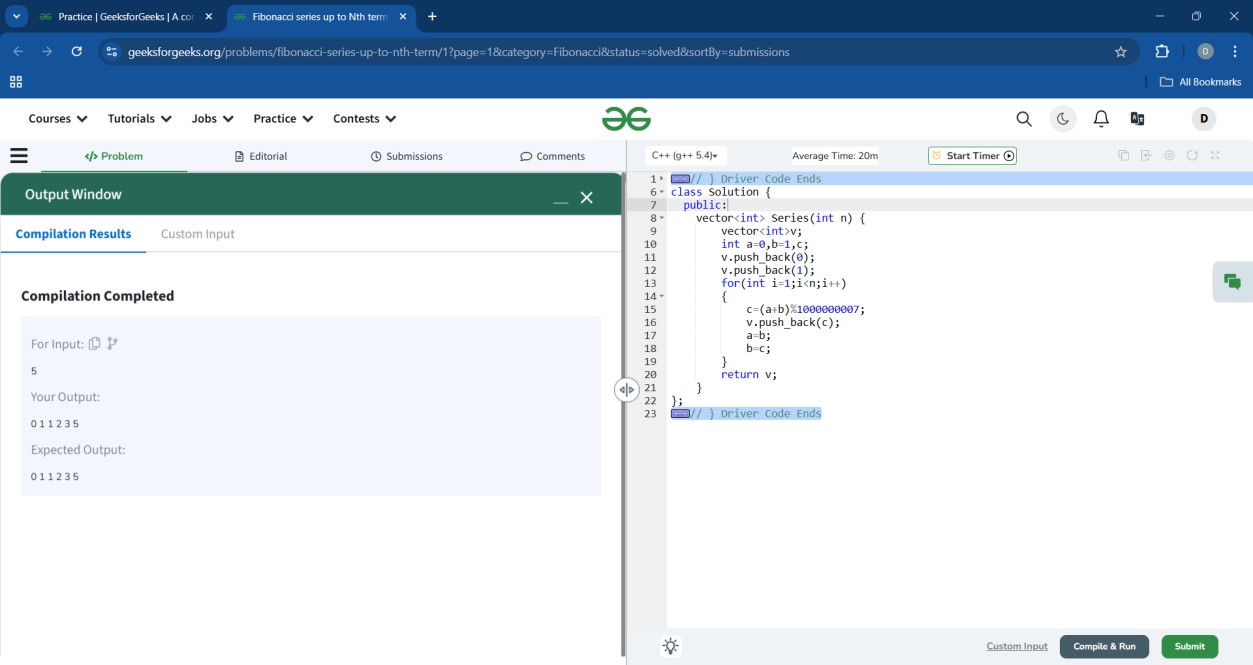
v.push\_back(c); a=b; b=c; } return v;

}

};

**3.**

**Fibonacci Sum**



class Solution{ public: long long int fibSum(long long int N){ long long int a=1,b=1,c,s=0; if(N==1) return a; else if(N==2) return a+b; else { s=2; for(int i=3;i<=N;i++)

{

c=a+b%1000000007; s=(s+c)%1000000007;

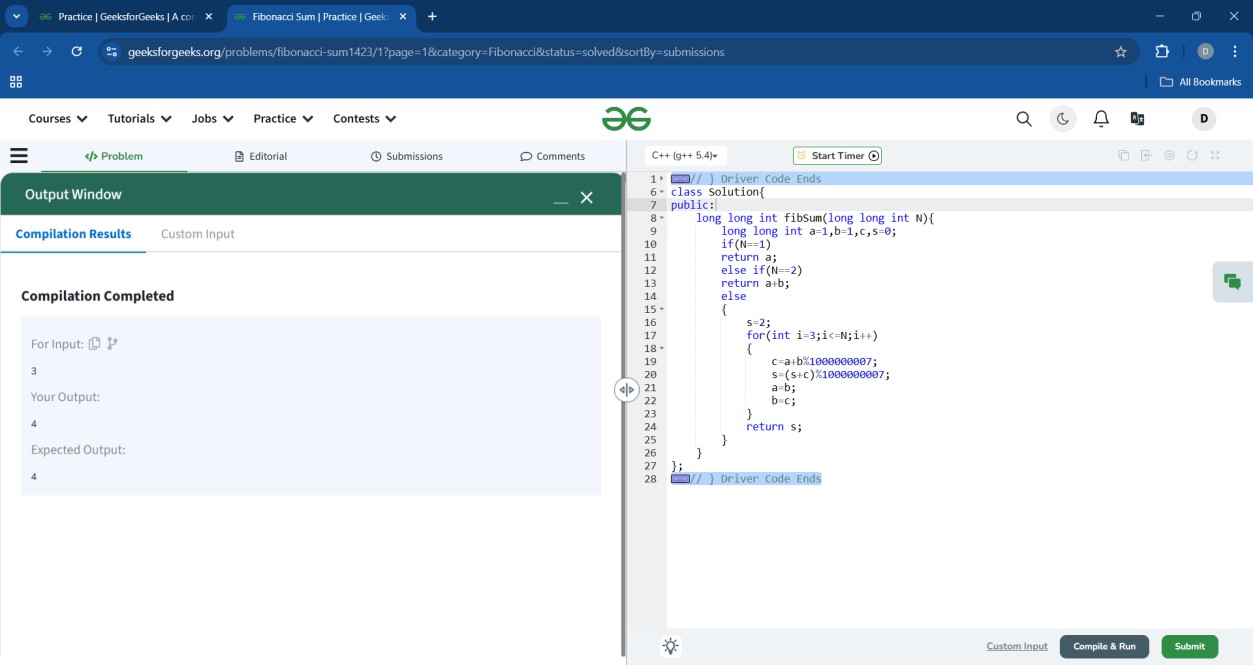
a=b; b=c; } return s;

}

}

}

;



## 4. Nth Even Fibonacci Number

class Solution { public:

long long int nthEvenFibonacci(long long int N) {

// code here long long int a=1,b=1,c,t=0; for(int i=3;i<=3\*N;i++)

{

c=(a+b)%1000000007;

a=b; b=c; } return b;

}

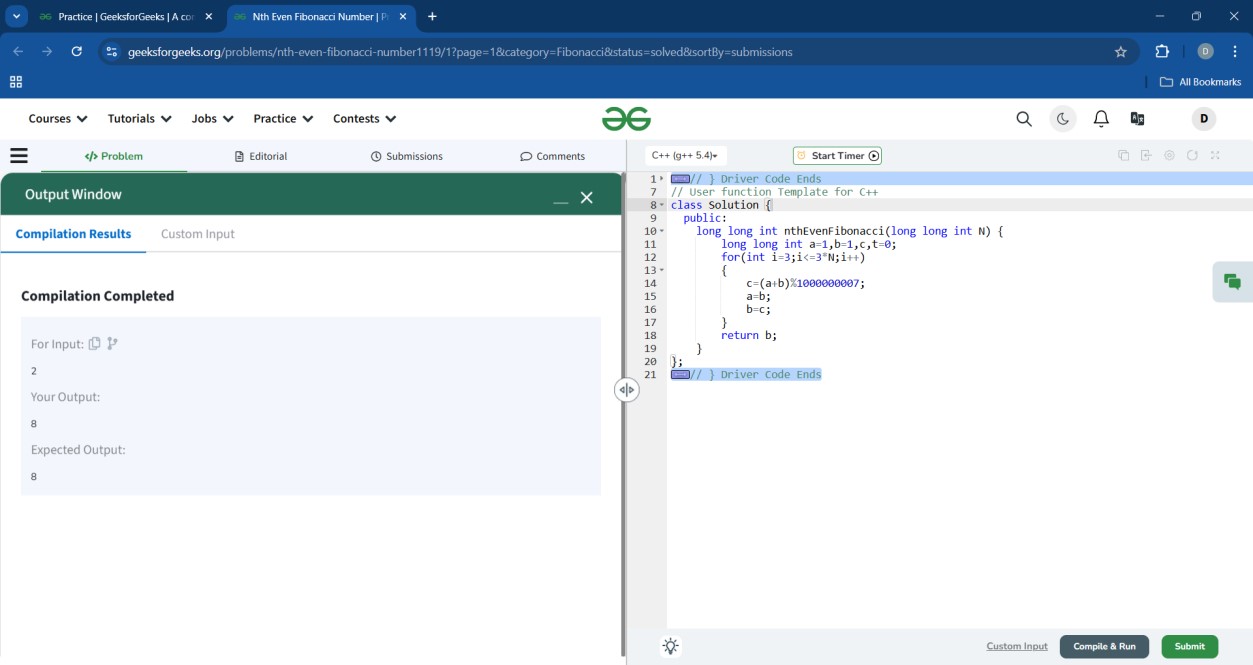
};

**5**

**.**

**c**

**heck if the number is Fibonacci**



class Solution{ public: bool isPerfect(int n)

{ int s=1,e=n; while(s<=e)

{ int m= floor(s+e)/2; if(m\*m==n) return true; else if(m\*m<n) s=m+1; else e=m-1; } return false;

}

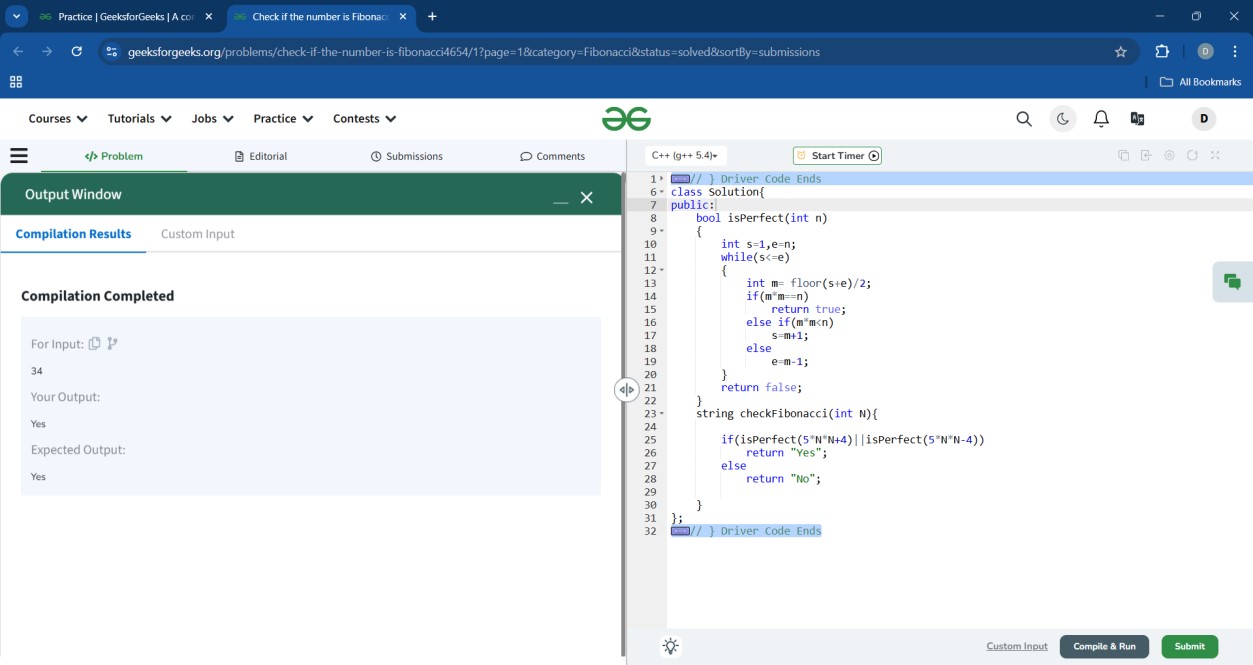
string checkFibonacci(int N){ if(isPerfect(5\*N\*N+4)||isPerfect(5\*N\*N-4))

return "Yes"; else return "No";

}

}

;



# General Series

## 1.Series AP

class Solution { public:

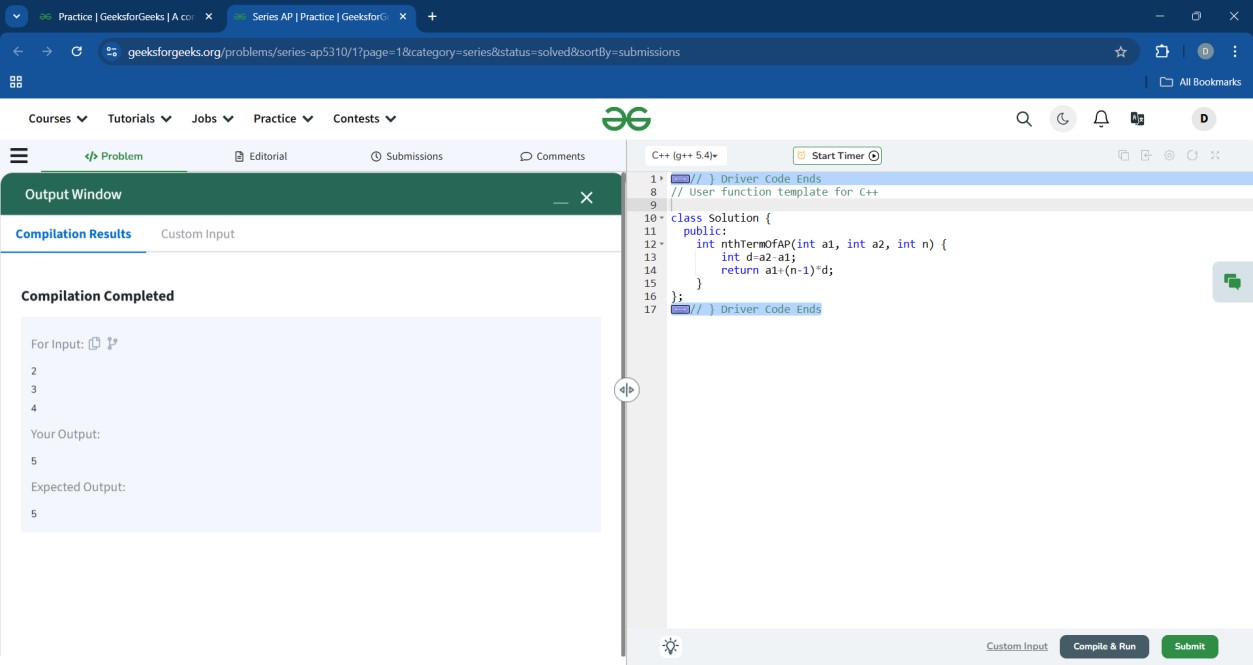
int nthTermOfAP(int a1, int a2, int n) { int d=a2-a1; return a1+(n-1)\*d;

}

};

**2**

**.Difference Series**



class Solution { public: int differenceSeries(int N)

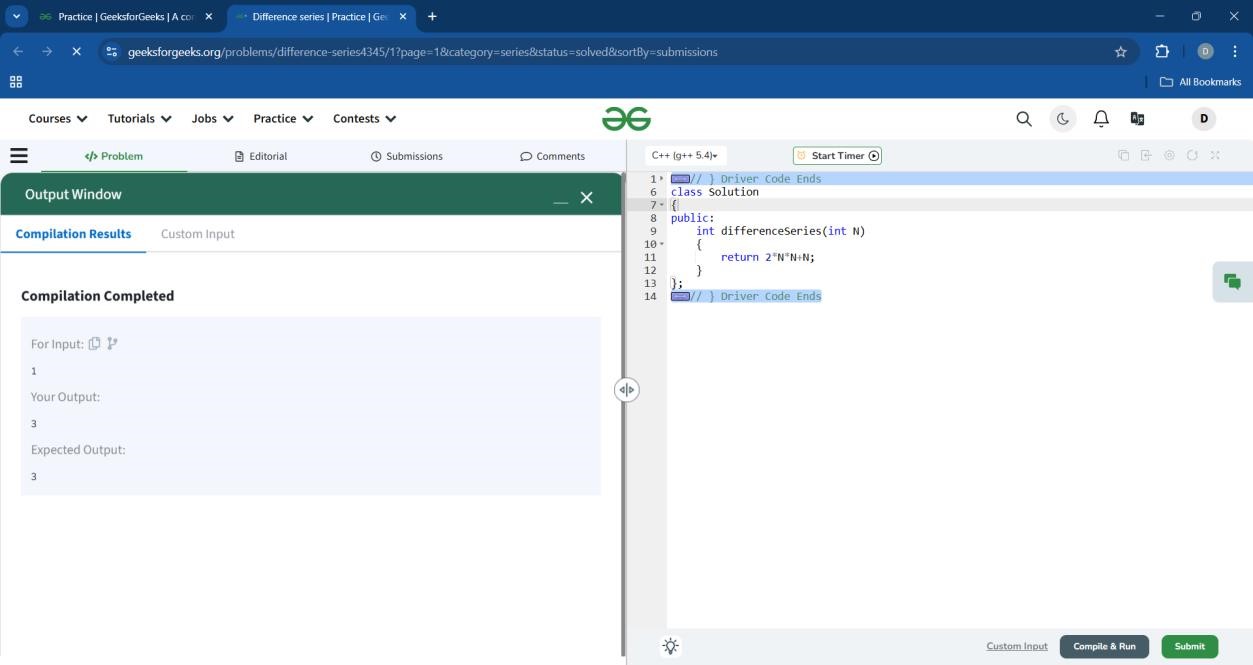
{

return 2\*N\*N+N;

}

}

;



## 3. Crack the series

class Solution { public:

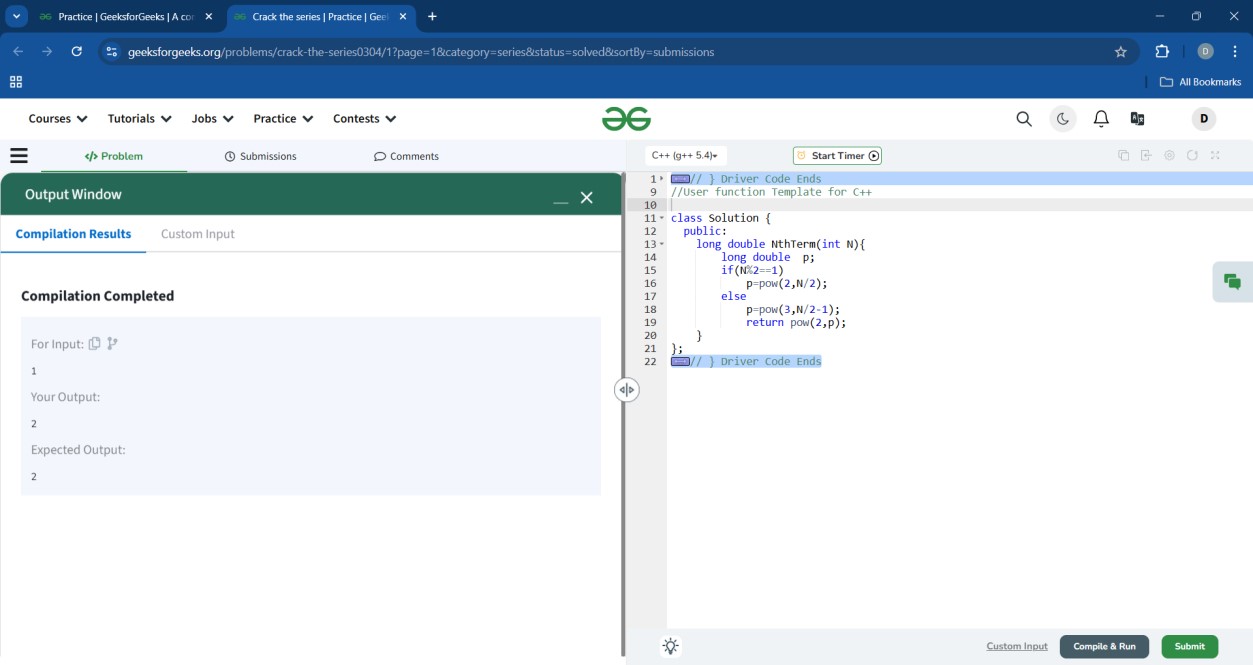
long double NthTerm(int N){ long double p; if(N%2==1) p=pow(2,N/2);

else p=pow(3,N/2-1); return pow(2,p);

}

}

;



## 4.Series X1

class Solution { public: long long X1Series(int X)

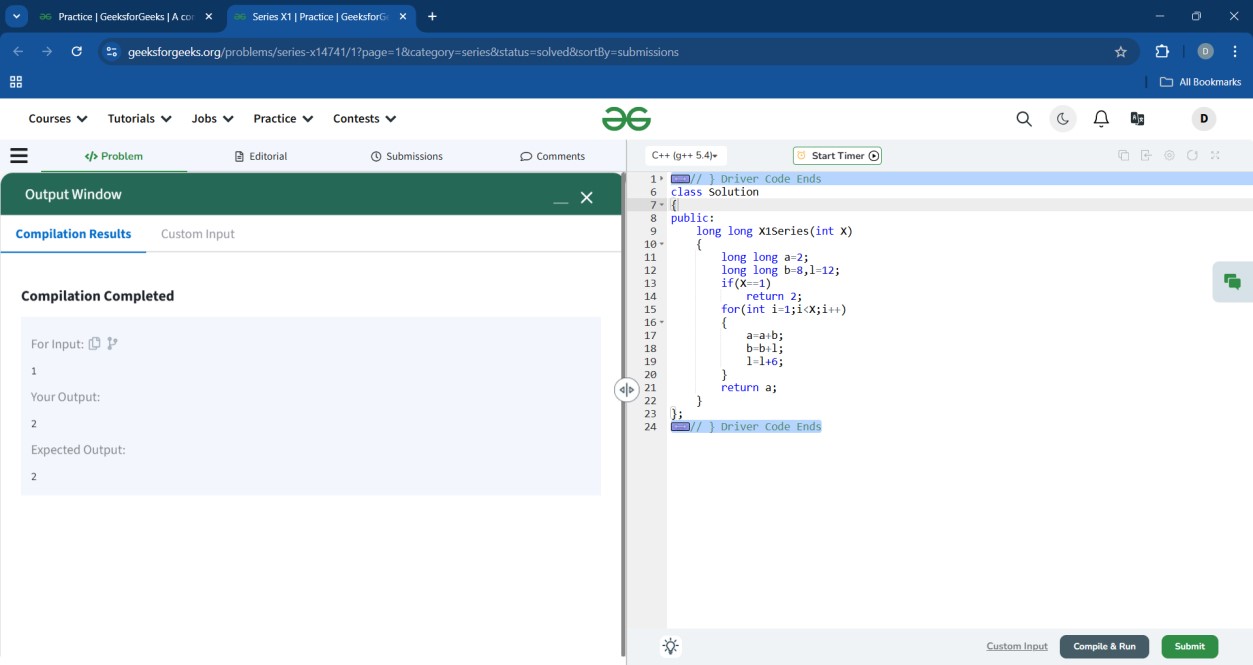
{ long long a=2; long long b=8,l=12; if(X==1) return 2; for(int i=1;i<X;i++)

{ a=a+b; b=b+l; l=l+6; } return a;

}

}

;



## 5. Mansi and her series

class Solution { public:

bool isPrime(int n){ for(int i=2;i<=sqrt(n);i++){ if(n%i==0){ return false;

} } return true; } int nthTerm(int n){

int i=0; int ans=0; for(int num=2;i<n;num++){ if(isPrime(num)){ ans=num; i++; }

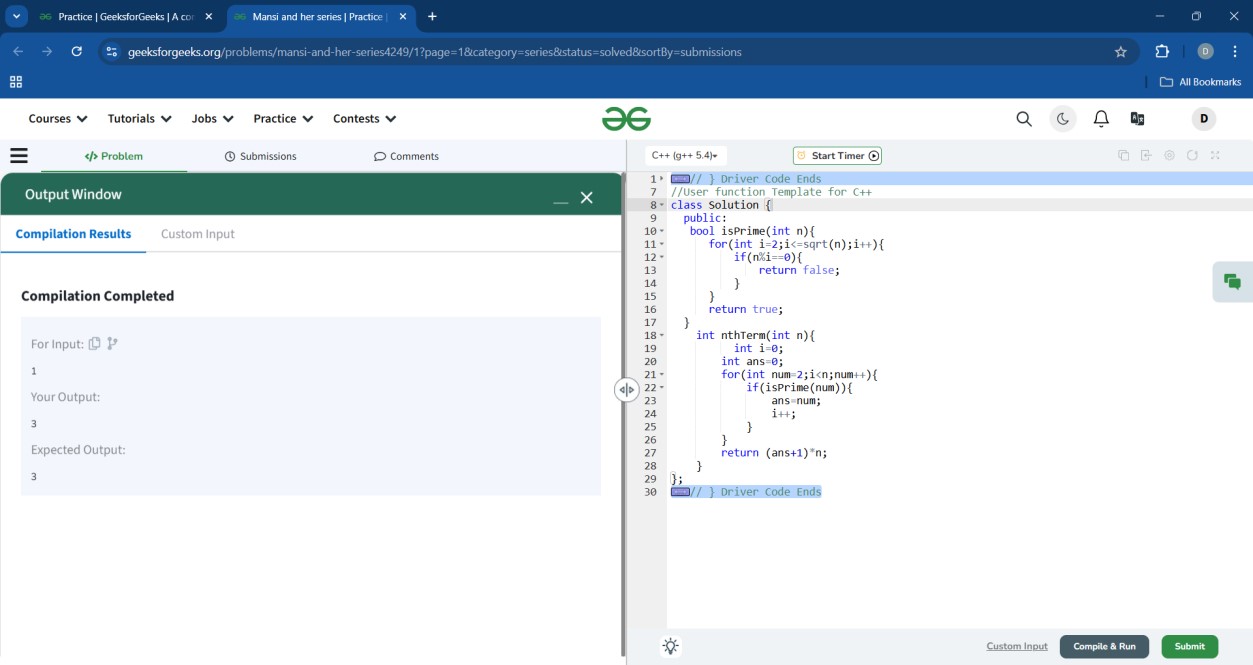
}

return (ans+1)\*n;

}

}

;



# Arrays

## 1.Missing in array

class Solution { int missingNumber(int arr[]) { int n = arr.length; int ans = 0; Arrays.sort(arr); if(n==1&&arr[0]<1){ ans = arr[0]+1; return ans;

} if(arr[0]==2){ return 1;

}

for(int i=0;i<n-1;i++){ if(arr[i]+1!=arr[i+1]){ ans = arr[i+1]-1; return ans;

}

}

return arr[n-1]+1;

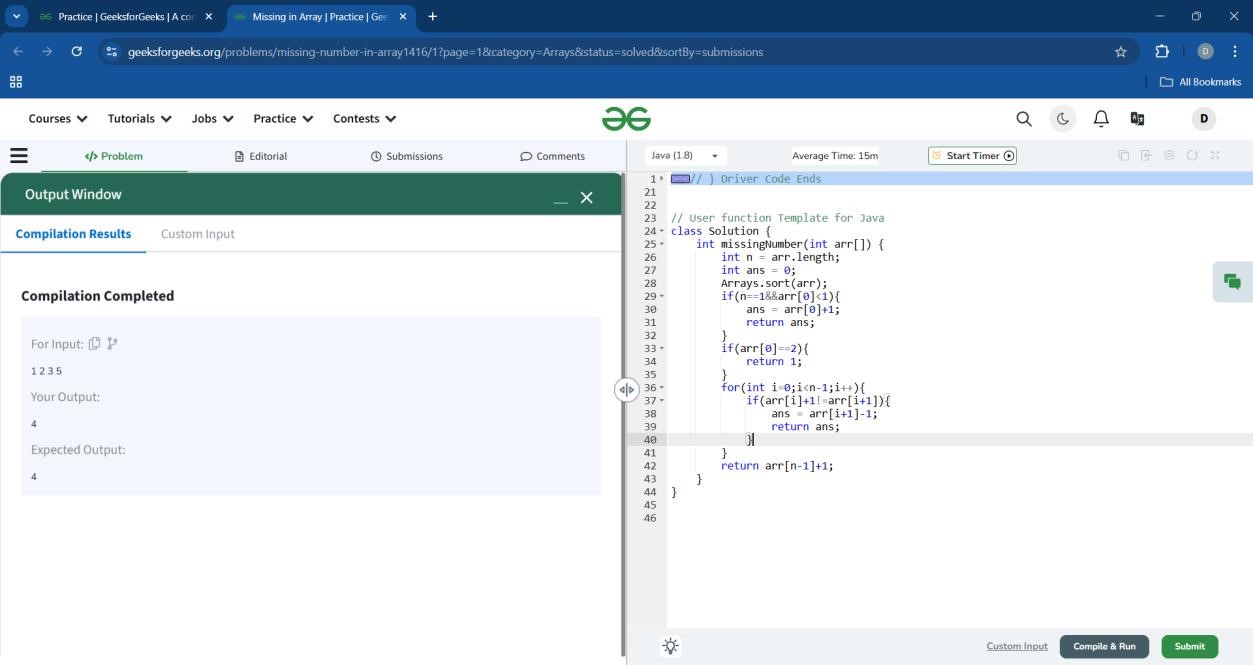
}

}

**2.**

**S**

**ort 0s, 1s and 2s**



class Solution { public void sort012(int[] arr)

{

int l=0,m=0,h=arr.length-1; while(m<=h)

{ if(arr[m]==0) { int temp=arr[l]; arr[l]=arr[m]; arr[m]=temp; l++; m++;

}

else if(arr[m]==1) { m++; } else { int temp=arr[h]; arr[h]=arr[m]; arr[m]=temp;

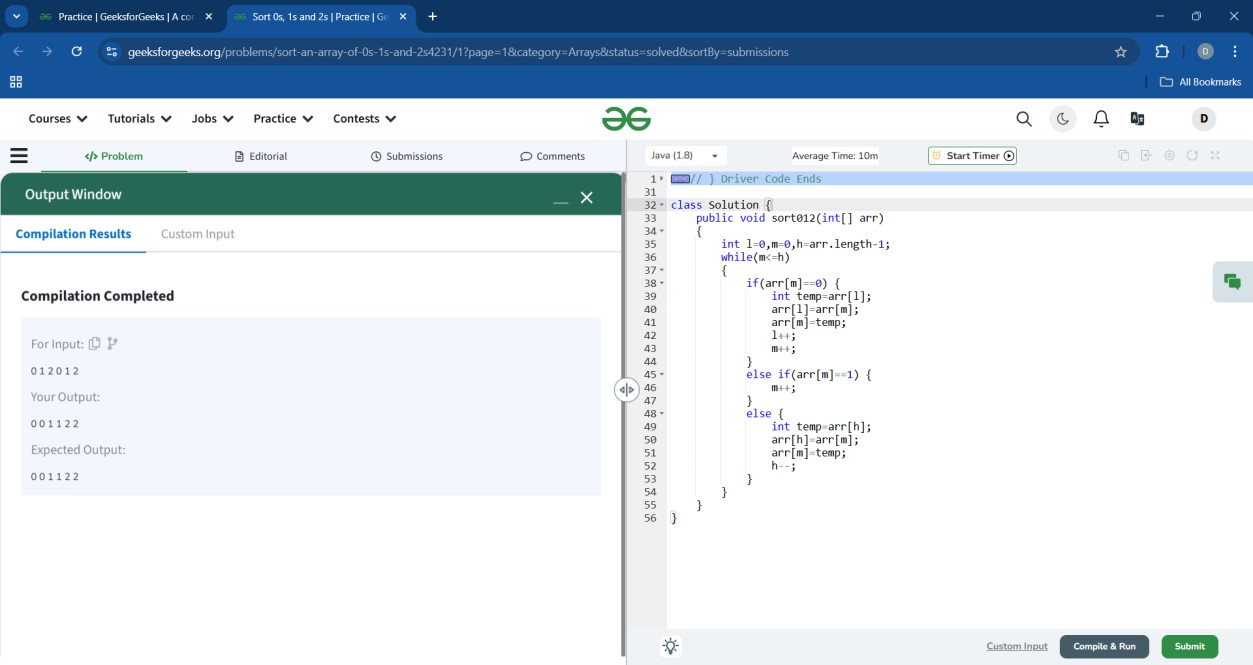
h--;

}

}

}

}



## 3.Binary Search

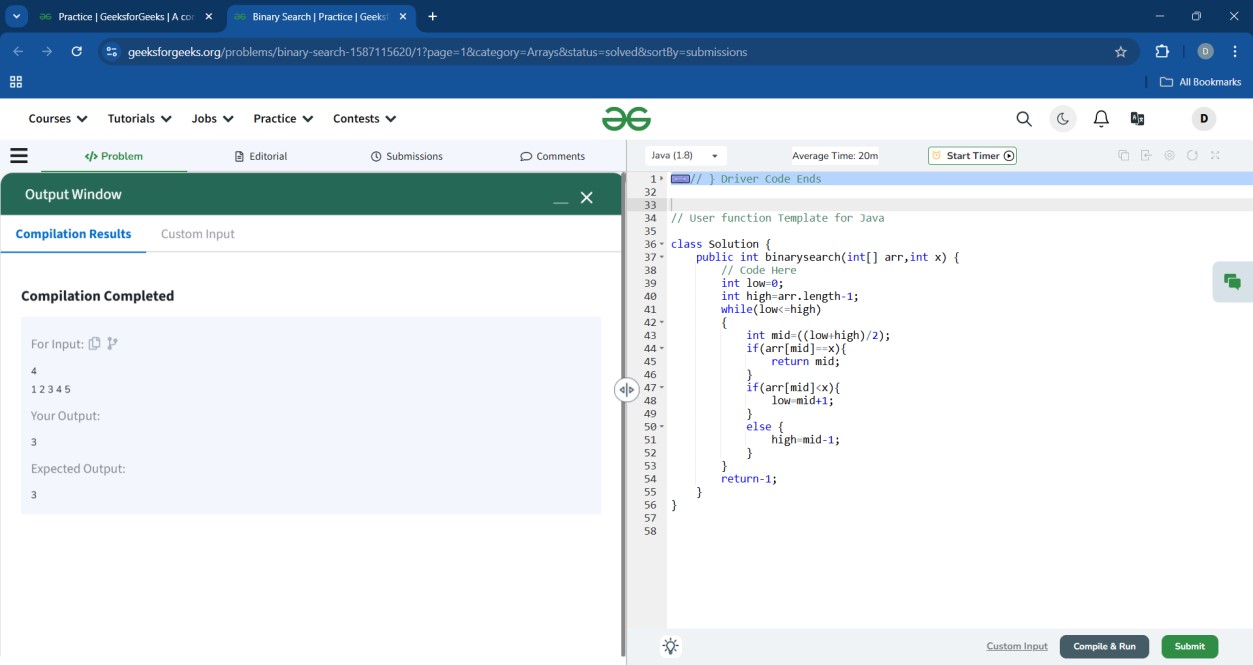
class Solution { public int binarysearch(int[] arr,int x) { int low=0; int high=arr.length-1; while(low<=high) { int mid=((low+high)/2); if(arr[mid]==x){ return mid;

} if(arr[mid]<x){ low=mid+1; } else { high=mid-1;

} } return-1;

}

}



## 4. Array Search

class Solution { static int search(int arr[], int x) {\

int i;

for(i=0;i<arr.length;i++)

{ if(arr[i] == x) { return i;

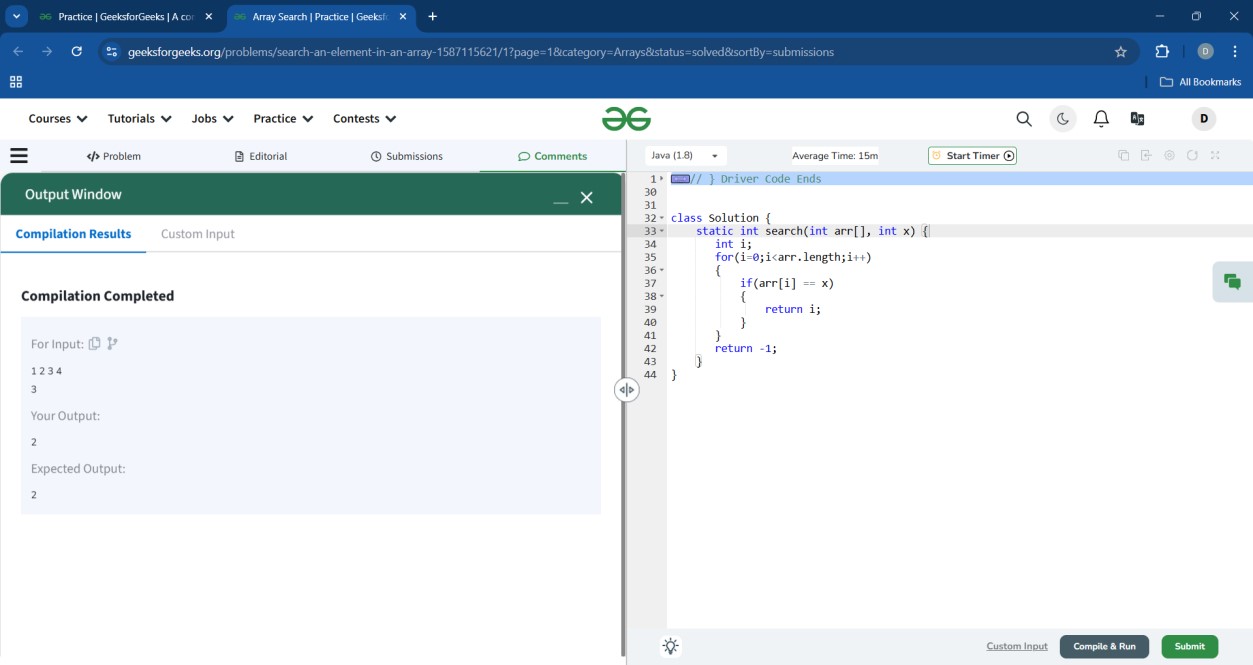
}

}

return -1;

}

}



## 5. Array Subset

class Compute { public String isSubset( long a1[], long a2[], long n, long m) { int count=0,i=0,j=0; Arrays.sort(a1);

Arrays.sort(a2); while(i<n&&j<m){ if(a1[i]==a2[j]){ count++; i++; j++;

}else if(a1[i]!=a2[j]){ i++;

}

}

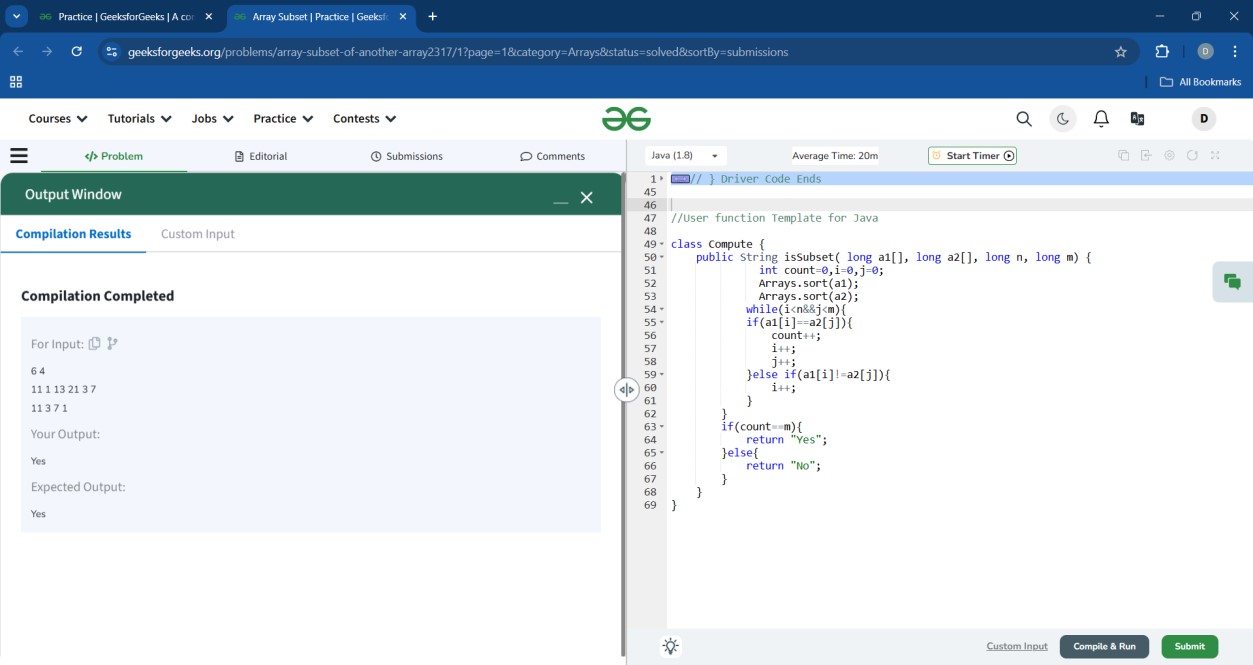
if(count==m){

return "Yes"; }else{ return "No";

}

}

}



## 6. Largest Element in Array

class Solution { public static int largest(int[] arr) { int max=0; for(int i=0;i<arr.length;i++)

{ if(arr[i]>max)

{ max=arr[i];

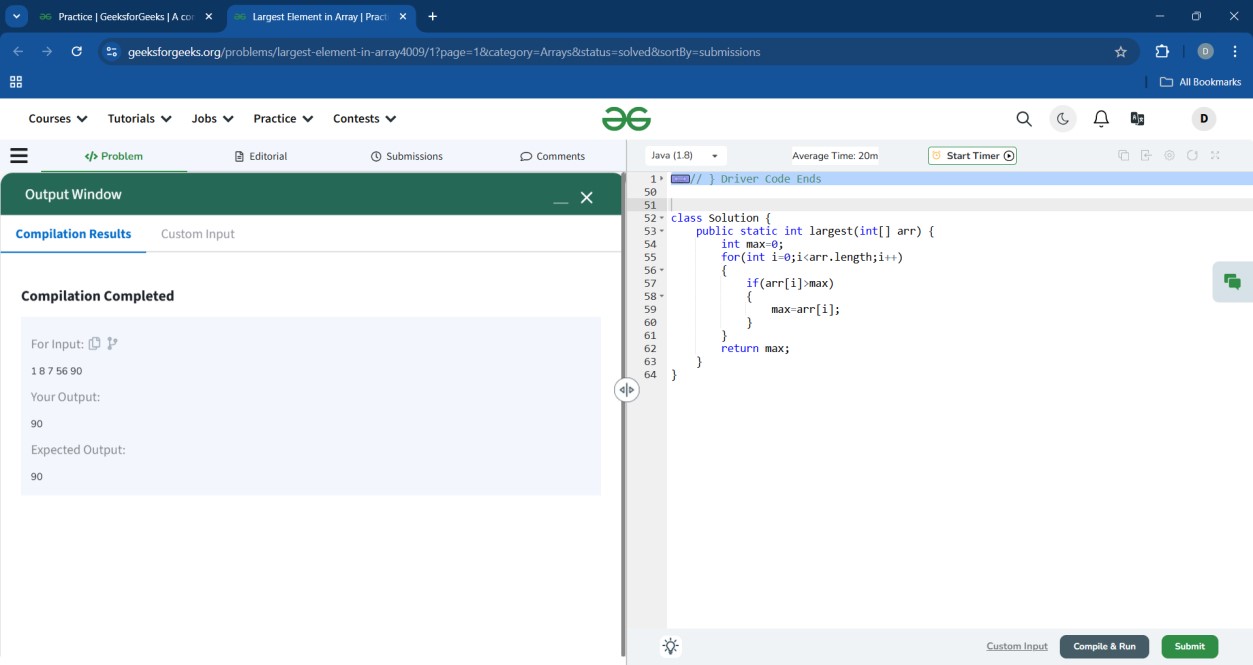
}

}

return max;

}

}



## 7. Rotate Array by One

class Solution { public void rotate(int[] arr)

{ int n = arr.length; if(n <= 1) { return;

} int temp = arr[n - 1]; for(int i = n - 1; i > 0; i--)

{ arr[i] = arr[i - 1];

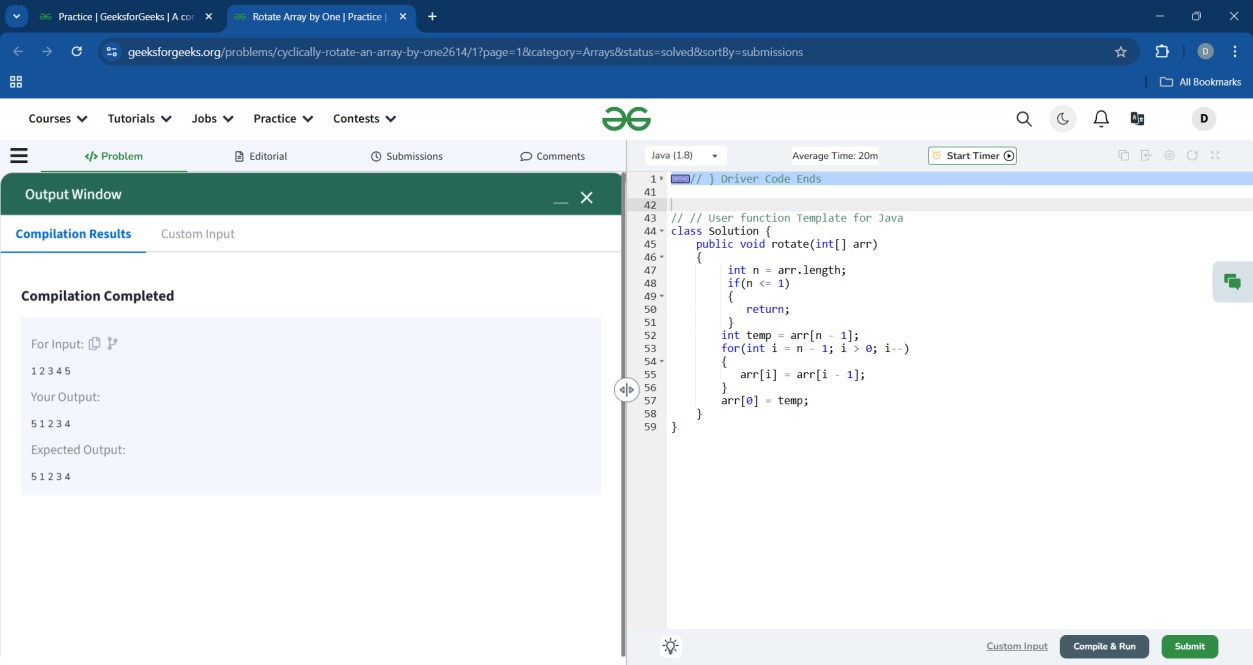
} arr[0] = temp;

}

}

**8.**

**Remove Duplicates Sorted Array**



class Solution { public int remove\_duplicate(List<Integer> arr) {

int n = arr.size(); int m = 0; for (int i=1;i<n;i++)

{ if (!arr.get(m).equals(arr.get(i)))

{ m++; arr.set(m,arr.get(i));

}

}

return m+1;

}

}

**9.**

**Wave Array**



class Solution {

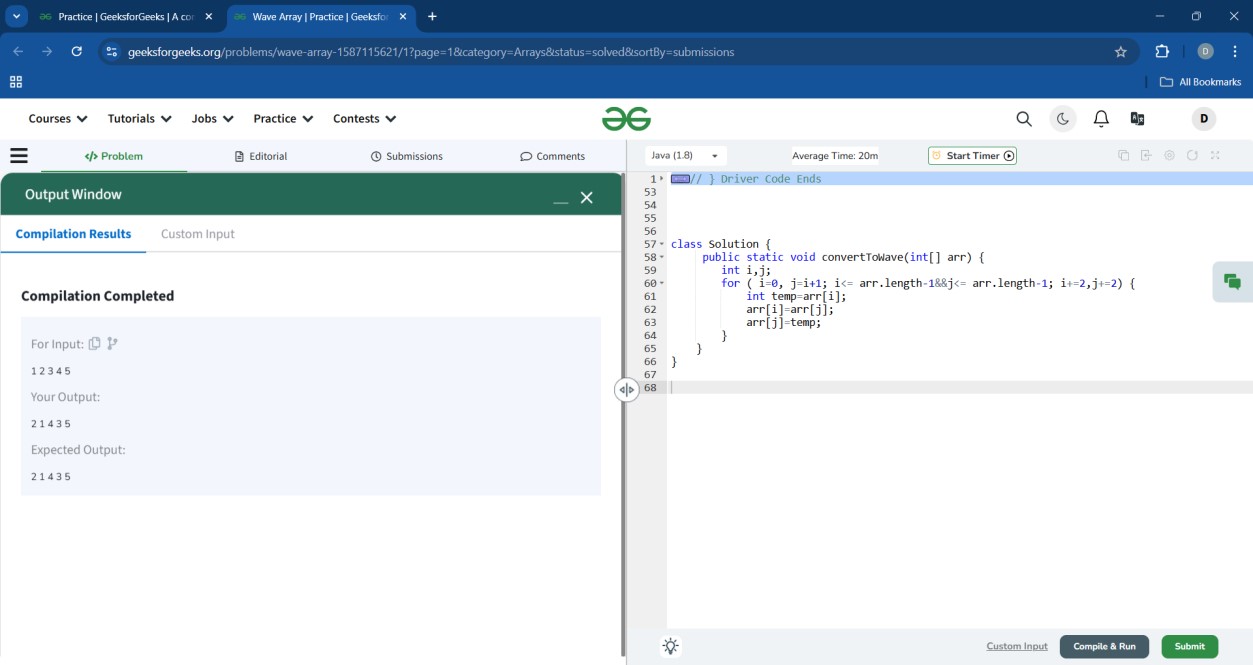
public static void convertToWave(int[] arr) { int i,j;

for ( i=0, j=i+1; i<= arr.length-1&&j<= arr.length-1; i+=2,j+=2) { int temp=arr[i]; arr[i]=arr[j]; arr[j]=temp;

}

}

}



## 10.Product array puzzle

class Solution { public static long[] productExceptSelf(int nums[]) { long res[]=new long[nums.length]; for(int i=0;i<nums.length;i++)

{ res[i]=1; for(int j=0;j<nums.length;j++)

{ if(i!=j)

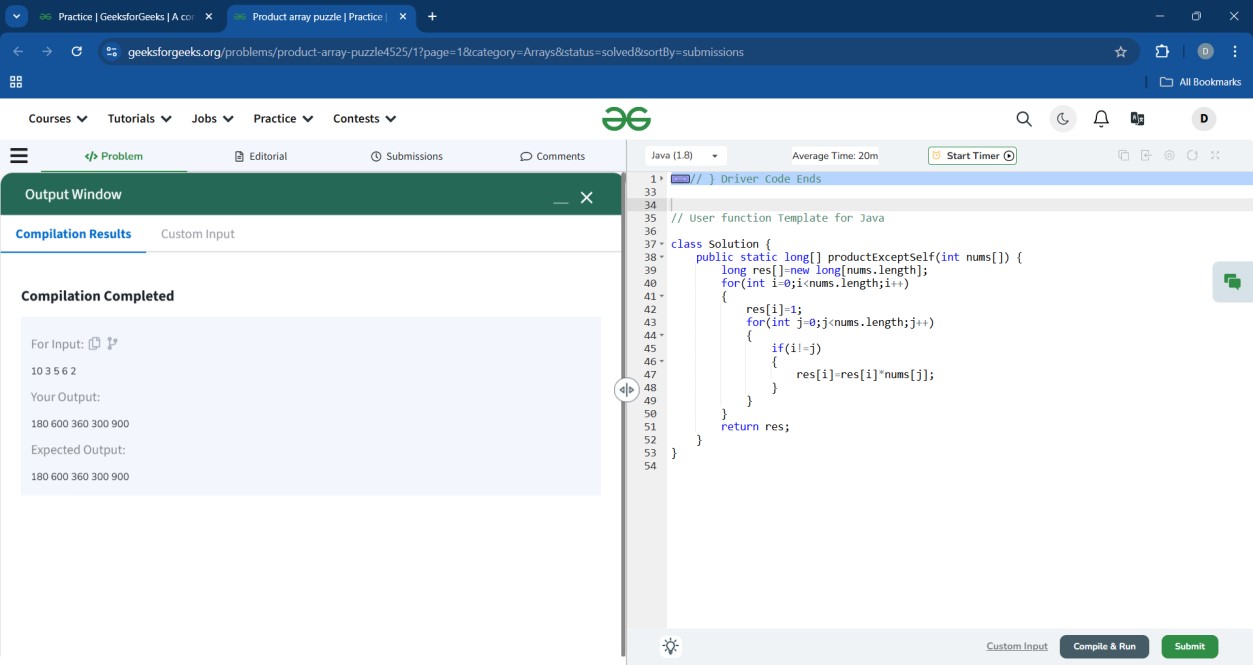
{ res[i]=res[i]\*nums[j];

}

} } return res;

}

}



## 11. Balanced Array

class Solution { public int minValueToBalance(List<Integer> arr) { int sum1=0,sum2=0;

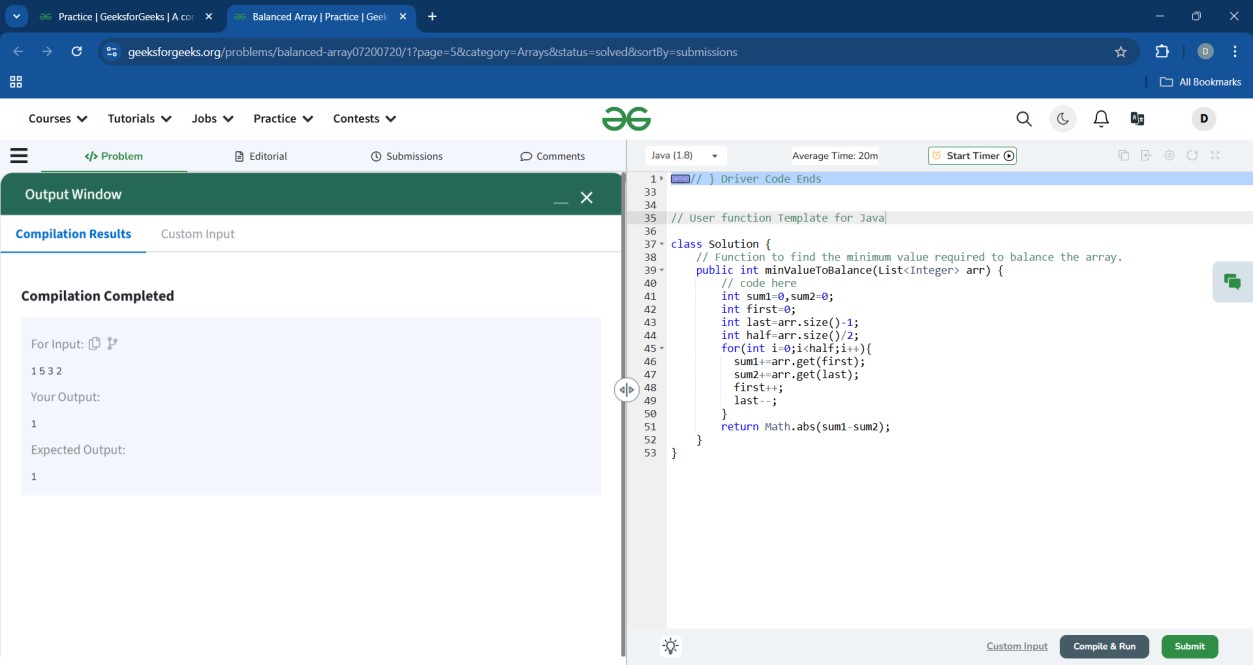
int first=0; int last=arr.size()-1; int half=arr.size()/2; for(int i=0;i<half;i++){ sum1+=arr.get(first); sum2+=arr.get(last); first++; last--;

}

return Math.abs(sum1-sum2);

}

}



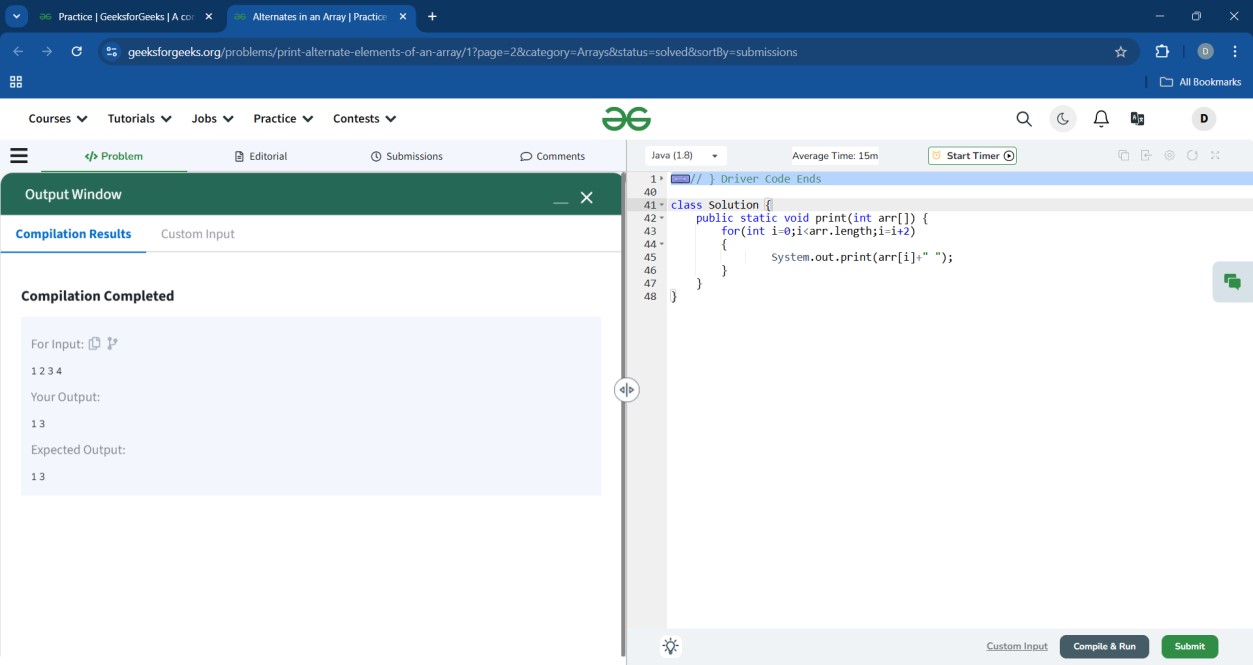
## 12.Alternates in an Array

class Solution { public static void print(int arr[]) { for(int i=0;i<arr.length;i=i+2) { System.out.print(arr[i]+" ");

}

}

}



## 13. Palindromic Array

class Solution { public static boolean palindrome(int n)

{ int mul=0,t=n; while(t!=0)

{

mul=mul\*10+t%10;

t=t/10;

}

if(mul==n)

{ return true; } else{

return false;

}

}

public static boolean palinArray(int[] arr) { for(int i=0;i<arr.length;i++)

{

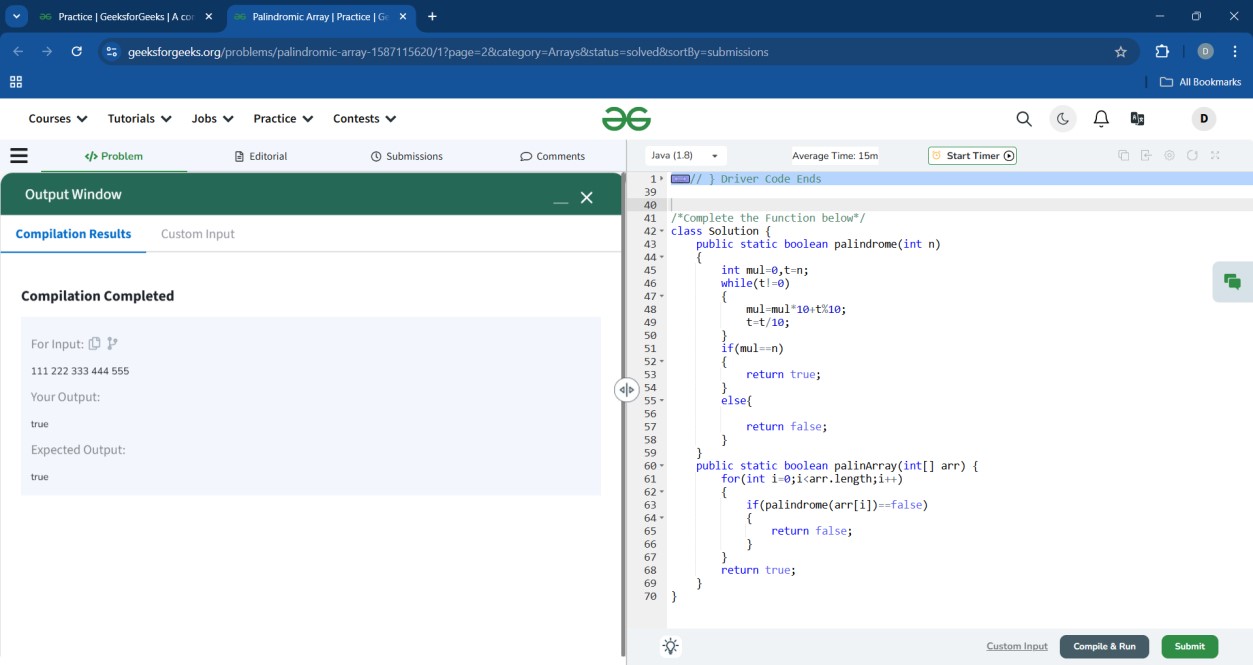
if(palindrome(arr[i])==false)

{ return false;

} } return true;

}

}



## 14. Adding Array Elements

class Solution { int minOperations(int[] arr, int n, int k)

{ int count = 0;

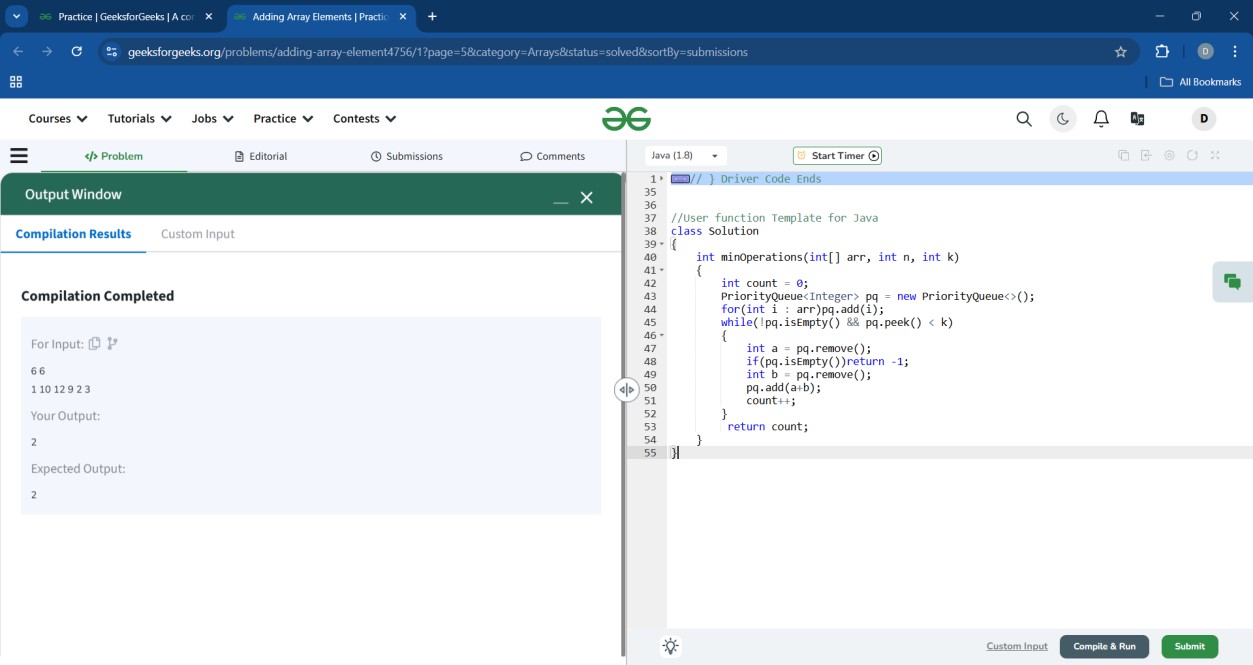
PriorityQueue<Integer> pq = new PriorityQueue<>();

for(int i : arr)pq.add(i); while(!pq.isEmpty() && pq.peek() < k)

{ int a = pq.remove(); if(pq.isEmpty())return -1; int b = pq.remove(); pq.add(a+b); count++; } return count;

}

**}**

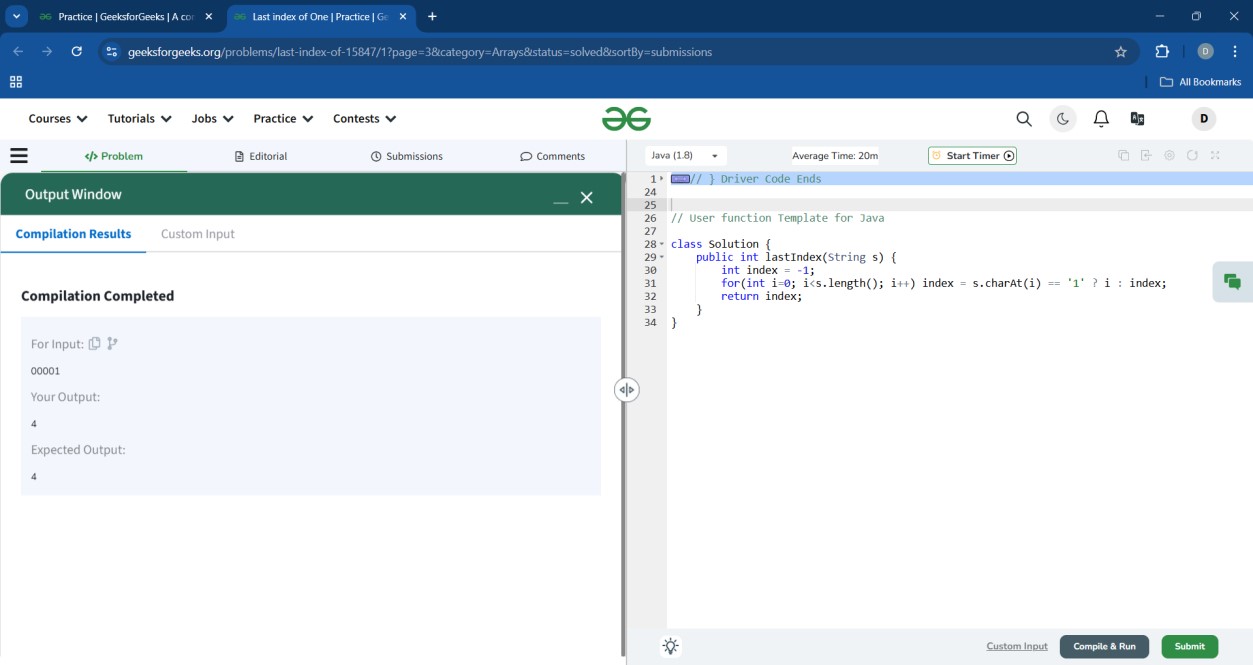


## 15.Last index of One

class Solution { public int lastIndex(String s) { int index = -1; for(int i=0; i<s.length(); i++) index = s.charAt(i) == '1' ? i : index; return index;

}

}



# Strings

## 1.Reverse a string

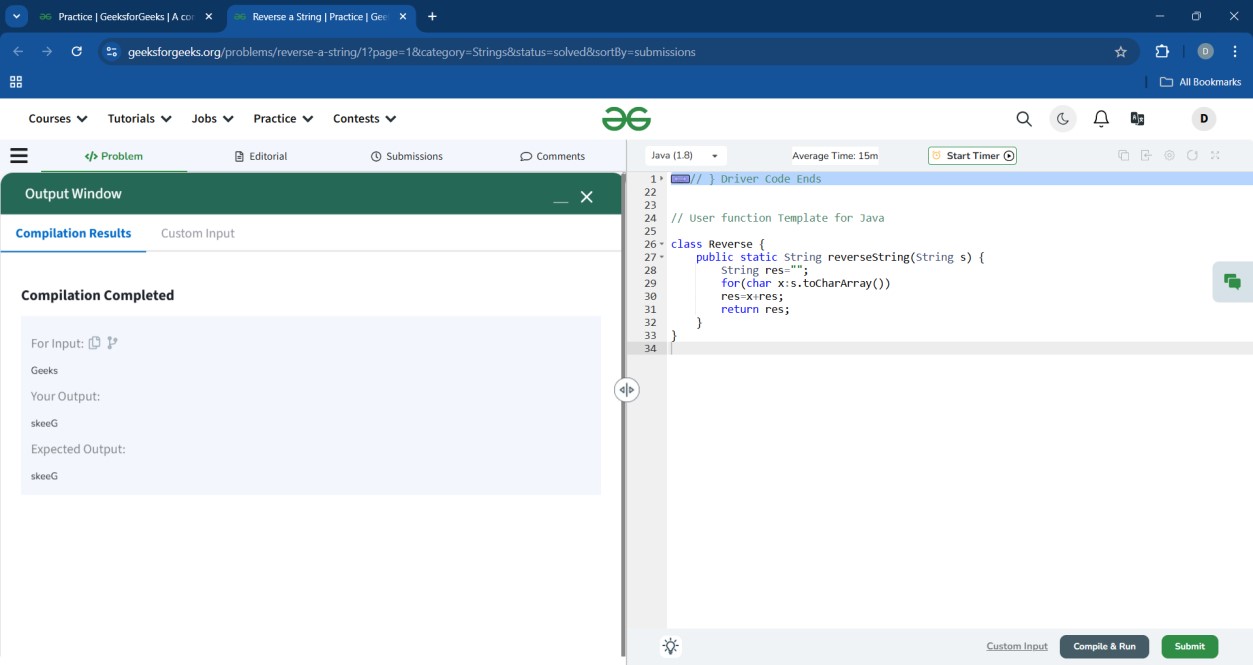
class Reverse { public static String reverseString(String s) { String res=""; for(char x:s.toCharArray()) res=x+res; return res;

}

}

**2.**

**Palindrome String**



class Solution { int isPalindrome(String S) {

String reversed = new StringBuilder(S).reverse().toString(); if(S.equals(reversed)){ return 1;

} else{ return 0;

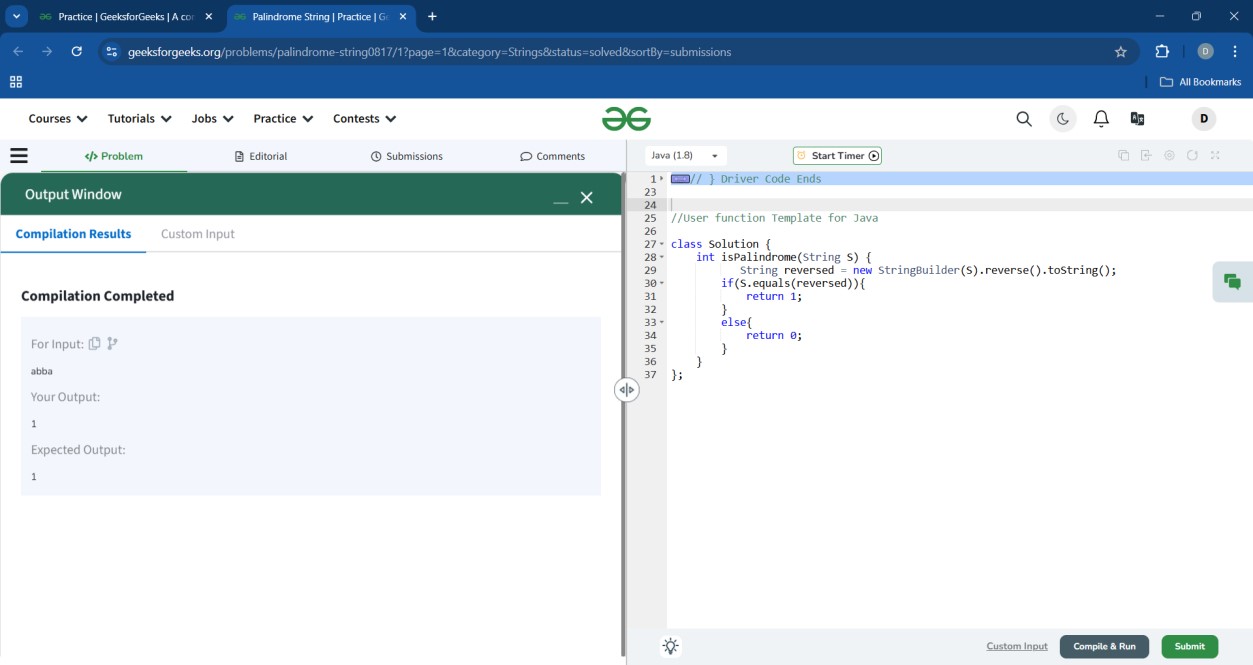
}

}

};

**3**

**.Reverse Words**



class Solution {

String reverseWords(String str) {

String[] words = str.split("\\.");

List<String> arr = Arrays.asList(words);

Collections.reverse(arr); String answer = String.join(".", arr); return answer;

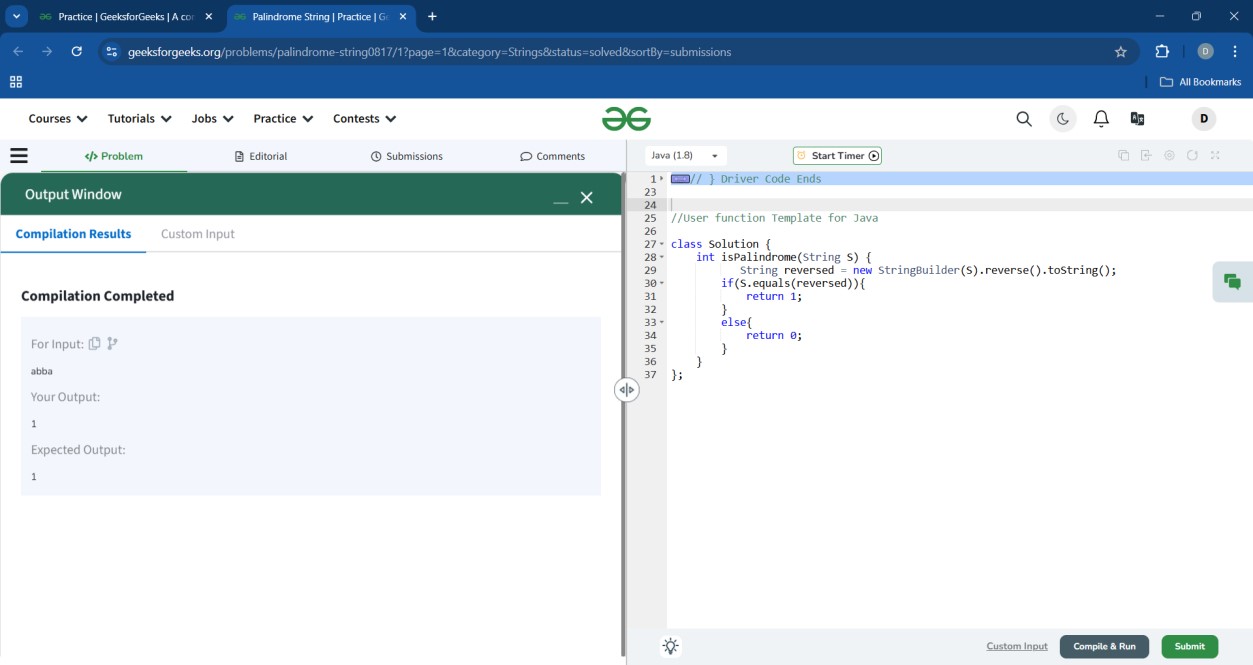
}

}

4

.

**Isomorphic Strings**



class Solution { public static boolean areIsomorphic(String str1,String str2)

{

HashMap<Character,Character>map=new HashMap<>();

if(str1.length()!=str2.length()) return false; int n=str1.length(); for(int i=0;i<n;i++)

{ char c=str1.charAt(i); if(map.containsKey(c))

{

if(map.get(c)!=str2.charAt(i)) return false;

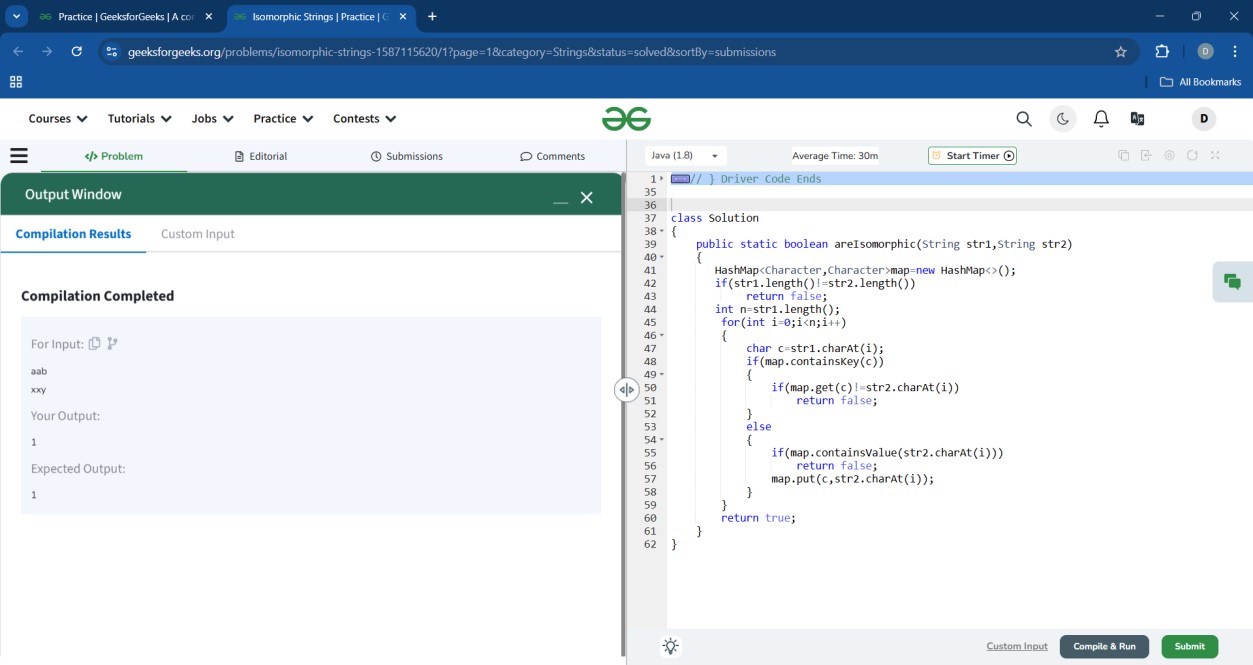
} else { if(map.containsValue(str2.charAt(i)))

return false; map.put(c,str2.charAt(i));

} } return true;

}

}



## 5. Implement strstr

class GFG{ int strstr(String s, String x)

{ int m=x.length(); int n= s.length(); for( int i=0;i<n-m+1;i++)

{

String s1=s.substring(i,i+m); if (s1.equals(x))

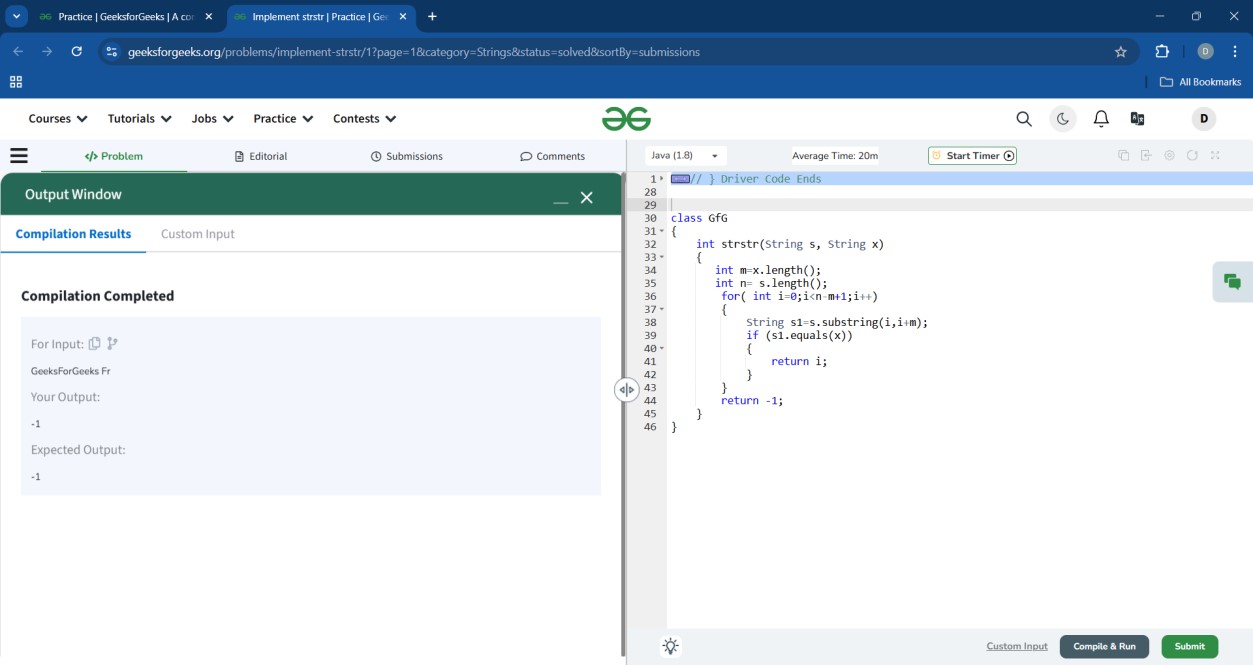
{

return i;

} } return -1;

}

}



## 6. Maximum Occuring Character

class Solution { public static char getMaxOccuringChar(String line)

{

TreeMap<Character,Integer> a=new TreeMap<>(); for(int i=0;i<line.length();i++)

a.put(line.charAt(i),a.getOrDefault(line.charAt(i),0)+1); int b=Integer.MIN\_VALUE;

char c=' '; for(Character x:a.keySet())

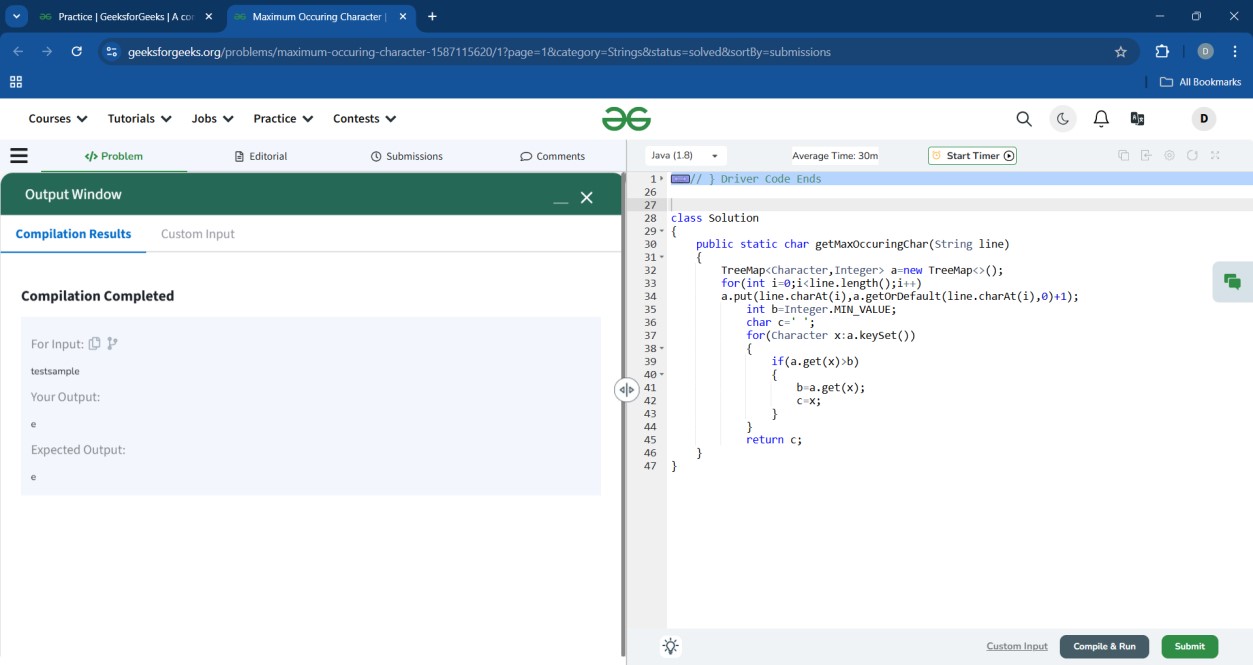
{ if(a.get(x)>b) {

b=a.get(x); c=x;

} } return c;

}

}



## 7. Remove Consecutive Characters

class Solution{ public String removeConsecutiveCharacter(String S){ String res=" "+S.charAt(0); for(int i=1;i<S.length();i++) { if(S.charAt(i-1)!=S.charAt(i)) res=res+S.charAt(i);

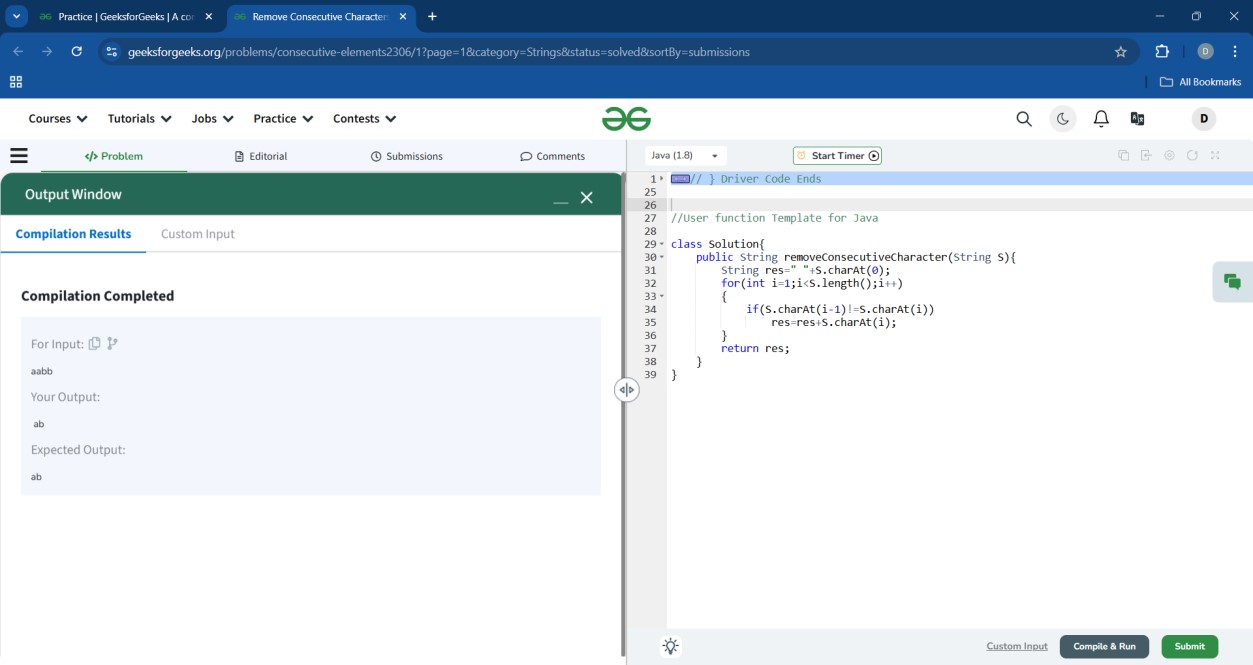
} return res;

}

}

**8.**

**Repeated Character**



class Solution { char firstRep(String S)

{

HashMap<Character,Integer> h=new LinkedHashMap<>();

for(int i=0;i<S.length();i++){ char ch=S.charAt(i);

h.put(ch,h.getOrDefault(ch,0)+1);

}

for(Character x:h.keySet())

{ if(h.get(x)>1) return x;

} return '#';

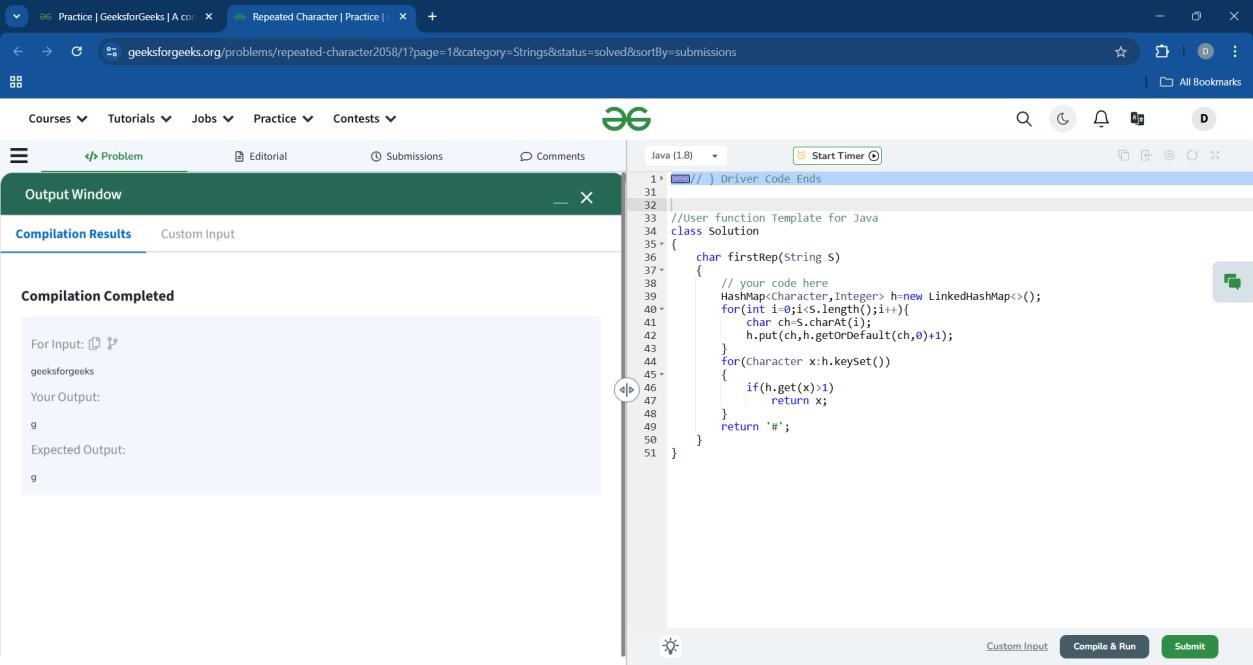
}

}

**9**

**.**

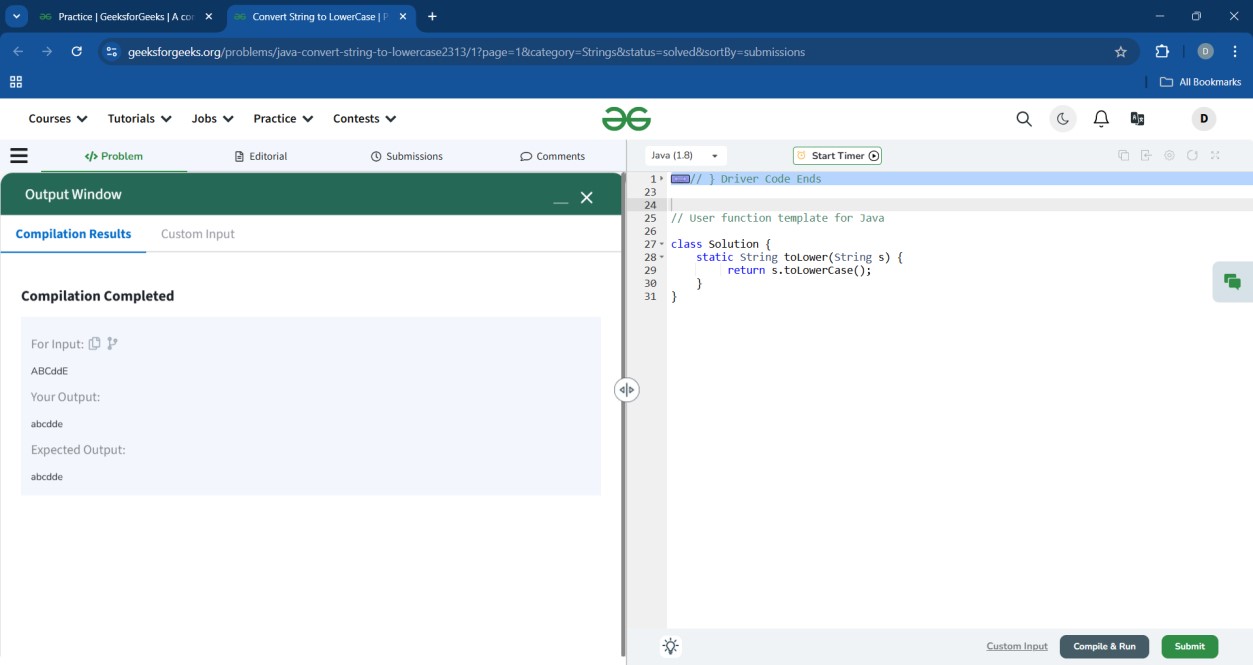
**Convert String to LowerCase.**



class Solution { static String toLower(String s) { return s.toLowerCase();

}

}



## 10. Display longest name

class Solution { public String longest(List<String> names) { String longestName = ""; int maxLength = 0; for (String name : names) { int length = name.length(); if (length > maxLength) { longestName = name; maxLength = length;

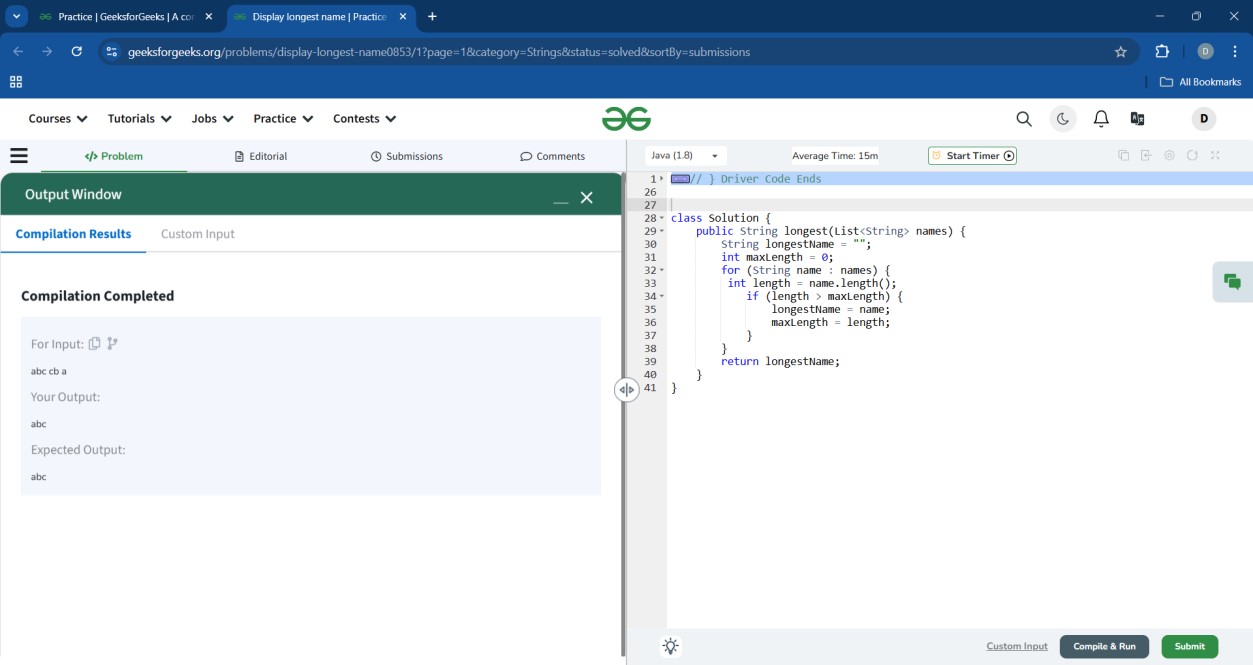
}

}

return longestName;

}

}



## 11. Remove common characters and concatenate

class Solution { public static String concatenatedString(String s1,String s2)

{

String str = ""; for(char ch : s1.toCharArray()) { if(s2.indexOf(ch) == -1) { str+=ch;

}

}

for(char ch : s2.toCharArray()) { if(s1.indexOf(ch) == -1) { str+=ch;

}

} return str.length() == 0 ? "-1" : str;

}

}



## 12. Sum of numbers in string

class Solution { public static long findSum(String str)

{ int sum = 0;

StringBuilder num = new StringBuilder(); for (int i = 0; i < str.length(); i++)

{ char ch = str.charAt(i); if (Character.isDigit(ch)) num.append(ch);

else

{ if (num.length()>0)

{

sum =sum+ Integer.parseInt(num.toString()); num.setLength(0);

}

}

}

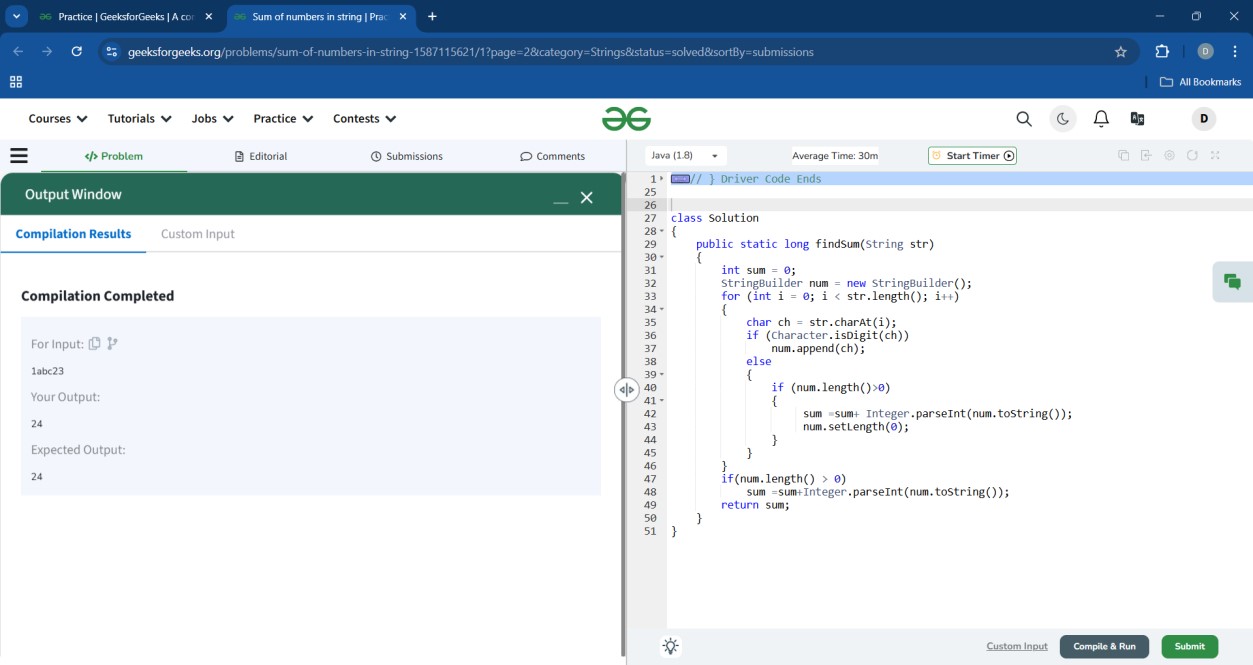
if(num.length() > 0) sum =sum+Integer.parseInt(num.toString()); return sum;

}

}

**13.**

**Java Strings | Set 1**



class Solution { static String conRevstr(String S1, String S2)

{

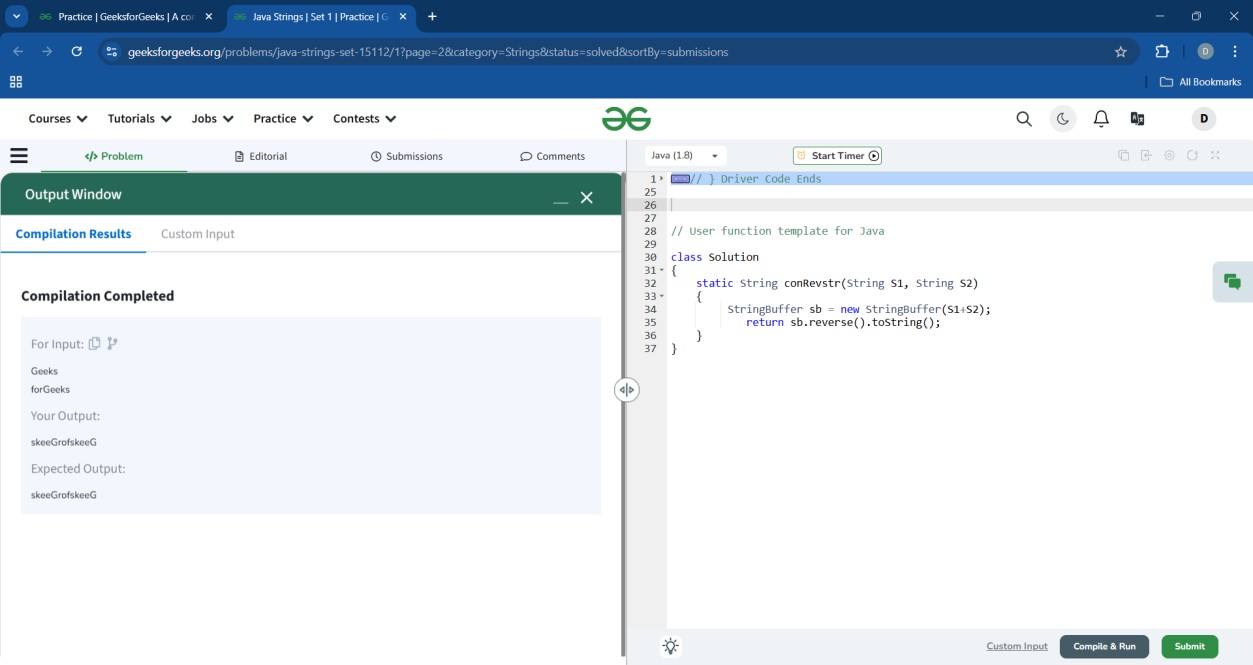
StringBuffer sb = new StringBuffer(S1+S2); return sb.reverse().toString();

}

}

**14.**

**C++ Strings**



class Sol

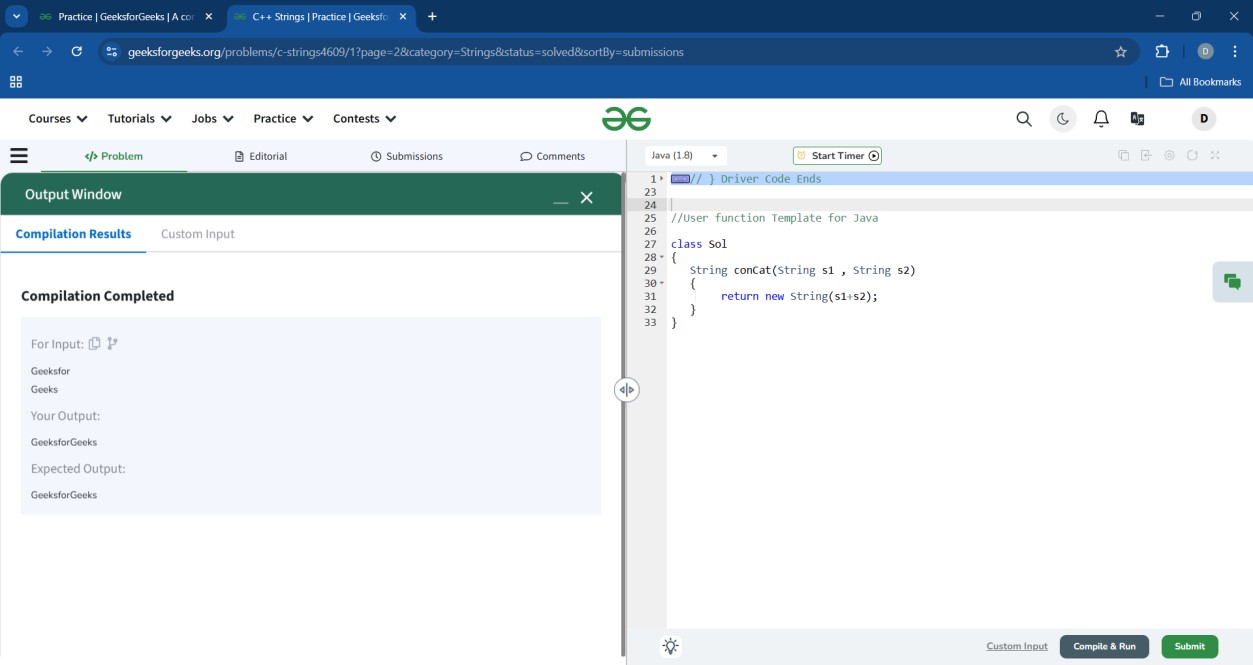
{

String conCat(String s1 , String s2)

{ return new String(s1+s2);

}

}



## 15. Pattern Searching

class Solution { int search(String text, String pat) { return (text.contains(pat))?1:0;

}

}

;

